Université catholique de Louvain
Ecole Polytechnique de Louvain
Institute of Information and Communication
Technologies, Electronics and Applied Mathematics

Multimodal and Nonsmooth Optimization on Matrix Manifolds

Doctoral Dissertation presented by

Pierre B. Borckmans

in partial fulfilment of the requirements
for the Ph.D. degree in Engineering Sciences

Thesis committee:

Profs.	Philippe Lefèvre,	(UCLouvain),	[President]
	Pierre-Antoine Absil,	(UCLouvain),	[Advisor]
	Paul Van Dooren	(UCLouvain),	[Advisor]
	Yurii Nesterov	(UCLouvain)	
	Moritz Diehl	(KULeuven)	
	Alain Sarlette	(UGent)	

Louvain-la-Neuve, August 2013

Foreword

When I started my PhD, nearly four years ago, it came as a surprise to many people around me. Admittedly, even to myself. Of course, I have always had an inclination for mathematics and scientific research but I explicitly stated many times during my studies that I would not follow this path. I had more practical ideas in mind and therefore started to work in a private company before taking this unexpected academic turn. *Never say never...* I am now nearing the end of my PhD adventure and I must say that completing this thesis has been an extremely enriching experience. Of course, I learned a lot about my research area topics but perhaps more importantly and less expectedly, I learned quite a lot about myself, my interests and my limitations. Like in any long term process, there were moments of excitement and surprise, but also moments of doubt and questioning. Fortunately, I was not alone in this journey and I could count on many people to orient me on this "definitely-not-so-smooth" path.

First, I would like to sincerely thank my advisors Pierre-Antoine Absil and Paul Van Dooren. I am especially grateful to Pierre-Antoine for his trust, his constant patience and understanding. Beyond these human qualities, I had the chance to enjoy his taste for scientific rigour, his expertise and his comforting guidance.

Another key ingredient to the success of this adventure is collaboration. I greatly appreciated working with Pierre-Antoine, Mariya Ishteva, Nicolas Boumal and Easter Selvan. Thank you all for the great interactions we had in writing these articles together.

And then there are those people with whom I did not have time/luck to co-author but who nevertheless interacted with me a lot. Because they made my experience better, richer and funnier, I owe big thanks to

Thomas Cason, Samuel Melchior, Arnaud Browet and Bastien Gorissen!

On a broader level, I enjoyed working in the Euler building. Between the frequent birthday pies, the passionate discussions during lunch, the numerous sport events and the nice atmosphere, there were always entertaining ways to escape briefly from the eternally ongoing research work.

As a F.R.I.A. research fellow, I thank the Fonds de la Recherche Scientifique (F.R.S.-F.N.R.S.) for their administrative and financial support. I also gratefully acknowledge the help from the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

I would also like to express my genuine appreciation to all the members of the jury–Philippe Lefèvre, Pierre-Antoine Absil, Paul Van Dooren, Yurii Nesterov, Moritz Diehl and Alain Sarlette–for devoting time and interest to the reading and evaluation of this manuscript.

I am especially grateful to Frédéric and Béatrice for welcoming us so warmly in Uzès, where most of this thesis was written. It has been a real delight to work in these fabulous circumstances...

Last but not least, my wonderful wife and my lovely daughter deserve an extraordinary mention. Diane, for your encouraging words and for bearing with me in moments of doubts and unavailability. Anouk, for finally sleeping through the night before I had to start the redaction of my thesis and for bringing so much joy to our family!

Abstract

Optimization is undoubtedly one of the most important fields of applied mathematics and as such received a lot of attention in the last decades. A tremendous amount of engineering applications are naturally expressed as optimization problems, or can easily be reformulated as such. Many classes of optimization problems are well identified nowadays and powerful solutions exist for a wide range of them. From linear programming and convex optimization to derivative-free optimization, from robust algorithms with in-depth convergence analysis to exotic search metaheuristics, the spectrum of available optimization methods is extremely large. For this reason, getting familiar with the optimization field can be overwhelming for the engineer who is looking for simple and effective solutions to a given problem. On the other hand, the pure theoretician is often dealing with highly specialized classes of optimization problems and he is mostly interested in providing convergence analysis. As a consequence of this admittedly caricatured dichotomy, it is not uncommon to observe practitioners making relatively poor choices of solvers. Furthermore, it is also quite rare to find optimization-related articles presenting a balanced trade-off between theoretical developments and more pragmatical approaches.

In this thesis, we tend to occupy this in-between research area. Through a series of well-known applications, originating from diverse engineering fields such as computer vision, machine learning and electrical power generation, we adopt a practical point of view and aim at designing ad-hoc solutions for these problems, based on advanced optimization techniques; at the same time, we endeavour to broaden the scope and propose developments that apply to larger classes of problems. To that end, the work presented in this thesis lies at the intersection of various optimization classes. The applications we study present

several challenges including piecewise smooth and non-differentiable objective functions, non-linearly constrained admissible sets, and multimodal landscapes. In order to cope with these difficulties, we resort to derivative-free techniques, exploration heuristics, optimization on manifolds and nonsmooth methods.

The main contribution of this thesis is in the development of specialized algorithms, combining techniques from diverse optimization fields, to produce adapted solutions to sophisticated problems. The geometrical structure of the search spaces is exploited using the framework of optimization on manifolds. We revisit existing optimization schemes, ranging from smooth optimization techniques to exploration heuristics, and adapt them to various manifold settings. Derivative-free techniques, such as the Mesh Adaptive Direct Search (MADS), and nonsmooth optimization methods, such as the subgradient descent, are also considered to cope with the non-differentiable aspects of some applications.

Finally, we present how concepts from standard optimization techniques can be combined with ideas coming from the meta-heuristics community to design hybrid solutions, providing global exploration of the search space, while ensuring robust local convergence properties.

Contents

List of Notations

Symbols

$a, b, \ldots, \alpha, \beta, \ldots$	Scalars
$\boldsymbol{a}, \boldsymbol{b}, \ldots, \boldsymbol{\alpha}, \boldsymbol{\beta}, \ldots$	Vectors
$\boldsymbol{A}, \boldsymbol{B}, \ldots$	Matrices
$\mathscr{A}, \mathscr{B}$	Tensors
$\boldsymbol{I}$	Identity matrix
a_{ijk}	Element of the tensor $\mathscr{A}$ at position (i, j, k)
$\boldsymbol{A}_{(n)}$	Mode-n matrix representation of tensor $\mathscr{A}$

Differential geometry

$\mathcal{M}$	Manifold
$\mathrm{T}_{\boldsymbol{x}}\mathcal{M}$	Tangent space at $\boldsymbol{x}$ of $\mathcal{M}$
$\mathcal{H}_{\boldsymbol{p}}\mathcal{M}$	Horizontal space at $\boldsymbol{x}$ of $\mathcal{M}$
$\mathcal{V}_{\boldsymbol{p}}\mathcal{M}$	Vertical space at $\boldsymbol{x}$ of $\mathcal{M}$
$\boldsymbol{\xi}, \boldsymbol{\eta}$	Tangent vectors
$P_{\boldsymbol{x}}(\boldsymbol{v})$	Projection of the vector $\boldsymbol{v}$ on the tangent space $\mathrm{T}_{\boldsymbol{x}}\mathcal{M}$
$R_{\boldsymbol{x}}(\boldsymbol{\xi})$	Retraction at $\boldsymbol{x}$ of the tangent vector $\boldsymbol{\eta}$
$\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{\xi})$	Exponential map at $\boldsymbol{x}$ of $\boldsymbol{\xi}$
$\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y})$	Logarithmic map at $\boldsymbol{x}$ of $\boldsymbol{y}$
$\Gamma_{\boldsymbol{p}}(\boldsymbol{\xi}, \boldsymbol{\eta})$	Parallel transport of the tangent vector $\boldsymbol{\eta} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ following the geodesic with initial velocity $\boldsymbol{\xi}$.
$\mathcal{T}_{\boldsymbol{\eta_p}}(\boldsymbol{\xi_p})$	Vector transport of the tangent vector $\boldsymbol{\eta_p} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ following the direction $\boldsymbol{\xi_p}$.

Sets

$\mathbb{N}$	The set of natural numbers
$\mathbb{Z}$	The set of integer numbers
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^n$	The set of real n-vectors
$\mathbb{R}^{m\times n}$	The set of $m \times n$ real matrices
$\mathbb{R}^{I_1\times I_2\times I_3}$	The set of $I_1 \times I_2 \times I_3$ tensors
$\mathrm{SO}(n)$	The special orthonormal group (set of rotation matrices)
$\mathfrak{so}(n)$	The set of skew-symmetric matrices
S^{n-1}	The unit hypersphere (set of all unit vectors in $\mathbb{R}^n$)
$\mathrm{St}(p,n)$	The Stiefel manifold (set of all column-wise orthonormal $(n \times p)$-matrices)
$\mathrm{Gr}(p,n)$	The Grassmann manifold (set of all p-dimensional subspaces of $\mathbb{R}^n$)
$\mathcal{OB}(p,n)$	The Oblique manifold (Cross product of p unit spheres)

Operations

$\boldsymbol{A}^\top$	Transpose of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of the matrix $\boldsymbol{A}$
$\det(\boldsymbol{A})$	Determinant of the matrix $\boldsymbol{A}$
$\mathrm{trace}(\boldsymbol{A})$	Trace of the matrix $\boldsymbol{A}$
$\otimes$	Kronecker product
$\oplus$	Direct sum
$\times$	Cartesian product
$\sum$	Sum
$\prod$	Product
$\mathrm{rank}(\cdot)$	Matrix rank
$\mathrm{rank}_n(\cdot)$	Mode-n tensor rank
$\lvert a\rvert$	Absolute value or module of the scalar a
$\lvert\mathcal{S}\rvert$	Cardinality of the set $\mathcal{S}$
$\mathrm{qf}(\cdot)$	Q-factor of the QR decomposition

Acronyms

PCA	Principal Component Analysis
ICA	Independent Component Analysis
OBB	Oriented Bounding Box
ELDP	Economic Load Dispatch Problem
VPE	Valve-point Effect
NM	Nelder-Mead
PSO	Particle Swarm Optimization
DE	Differential Evolution
MADS	Mesh Adaptive Direct Search
SVD	Singular Value Decomposition
HOSVD	Higher-Order Singular Value Decomposition
SD	Steepest Descent

Miscellaneous

$\text{co}\{\cdot\}$	Convex hull
$\mathcal{O}(\cdot)$	Big O complexity
e.g.	For example (*exempli gratia*)
i.e.	That is (*id est*)

List of Figures

List of Publications

Journal Articles

[BCA13] S. Easter Selvan Pierre B. Borckmans, Amit Chattopadhyay, and P.-A. Absil. Spherical mesh adaptive direct search for separating quasi-uncorrelated sources by range-based independent component analysis. *Neural Comput.*, 2013. accepted.

[BSBA13] Pierre B. Borckmans, S. Easter Selvan, Nicolas Boumal, and P.-A. Absil. A Riemannian subgradient algorithm for economic dispatch with valve-point effect. *J. Comput. Applied. Math.*, 2013. accepted.

Peer-reviewed Conference Articles

[BA10] Pierre B. Borckmans and Pierre-Antoine Absil. Oriented bounding box computation using particle swarm optimization. In *Proceedings of the 18th European Symposium on Artificial Neural Networks*, ESANN 2010, 2010.

[BIA10] Pierre B. Borckmans, Mariya Ishteva, and Pierre-Antoine Absil. A modified particle swarm optimization algorithm for the best low multilinear rank approximation of higher-order tensors. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 13–23, Berlin, Heidelberg, 2010. Springer-Verlag.

Conference Abstracts

[BA10a] P.B. Borckmans and P.-A. Absil. Fast oriented bounding box computation using particle swarm optimization. In *Proceedings of the 24th Annual Conference of the Belgian Operational Research Society (ORBEL)*, Liege, Belgium, January 28–29 2010.

[BA10b] P.B. Borckmans and P.-A. Absil. Fast oriented bounding box computation using particle swarm optimization. In *Book of Abstracts of the 29th Benelux Meeting on Systems and Control*, Heeze, Netherlands, March 2010.

[PBBA12a] N. Boumal P. B. Borckmans, E.S. Selvan and P.-A. Absil. A geometric subgradient descent algorithm for the economic load dispatch problem. In *Proceedings of the 20th International Symposium on Mathematical Theory of Networks and Systems*, Melbourne, Australia, July 9–13 2012.

[PBBA12b] N. Boumal P. B. Borckmans, E.S. Selvan and P.-A. Absil. A geometric subgradient descent algorithm for the economic load dispatch problem. In *Book of Abstracts of the 31th Benelux Meeting on Systems and Control*, Heijen, Netherlands, March 2012.

Chapter 1

Introduction

1.1 General introduction

In our everyday life, we make extensive use of optimization tools, most of the time without being aware of it. We no longer worry about getting lost: we use a GPS system in our car or a map engine before leaving home (for the unlucky ones). We surf the Internet using the Google web search engine, dozens of times a day. The class schedules for large schools and universities are automatically generated, dealing with hundreds of (sometimes incompatible) personal requirements. We buy and use products of all sorts that were built in highly optimized production factories, in order to lower the selling prices. Our economy system is partly based on portfolio management and optimization. And these are just a few examples in which efficient optimization tools play a central role.

Optimization, also referred to as mathematical programming, is undoubtedly one of the most important fields of applied mathematics. An endless amount of engineering applications are naturally expressed as optimization problems, or can easily be reformulated as such. Given the importance and the practical need of finding high quality and/or fast solutions to these problems, there is no wonder why the optimization field has received such attention in the last decades. However, a striking observation is that optimization problems arise in such diverse areas—ranging from econometrics, through industrial processes, to pure mathematics—that the meaning of the word optimization itself might differ from community to community. For the mathematician, an optimization (class of) problem is of great interest if theoretical results can

be derived. Quite often, the analysis carried out in this area heavily relies on properties and assumptions that fail to hold in real-life situations. For the pragmatic engineer on the other hand, optimization is often seen as an everyday tool and in that regard it should be easy to use and to tune. Here, the practical efficiency of the methods prevails over theoretical results and this philosophy sometimes leads to poor decisions regarding the choice of a suitable solver.

In this thesis, we aim at throwing some bridges between these two main communities. That is not to say that this in-between area is a no man's land and the description of the two extreme opposite camps presented above is of course quite caricatured. However, it is relatively rare in the literature to observe a well balanced trade-off between theoretical results and pragmatical solutions. More precisely, it is common to see engineers resorting to heuristics to solve a specific problem at hand, based on the fact that the problem is non standard. By doing so, they may miss the opportunity to enjoy convergence properties offered by classical optimization methods, because these do not seem to be applicable at first glance. The same optimization problem might be overlooked by the theoretician, claiming that it is either badly formulated or too difficult to obtain interesting results. A particularly interesting illustration of this situation is given by one of the applications treated in this thesis, namely the Economic Load Dispatch Problem (covered in Chapter 7). While being a reasonably simple problem to state mathematically, it appears that no real careful analysis of the problem was done in the power systems community, while on the other hand, dozens of articles proposing more and more exotic optimization heuristics are getting published, justified by the fact that the problem is not suitable for classical optimization techniques.

In our work, we try to reconcile the theoretical and the pragmatical points of view. To that end, a series of well known applications are visited, originating from diverse engineering fields such as computer vision, linear algebra, machine learning and electrical power generation. An overview of these applications can be found in Chapter 2. These applications present challenging characteristics: the corresponding objective functions are *non-convex*, *multimodal* and even *non-differentiable* (*nonsmooth*) in some cases. Also, the associated search spaces are *non-linearly constrained*.

The approach presented in this thesis is progressive. Each of the precited challenges is considered in turn and practical solutions are proposed

to tackle them. In order to develop adequate methods for the studied applications, the work presented in this thesis lies at the intersection of various optimization fields, as ingredients from diverse research areas are needed.

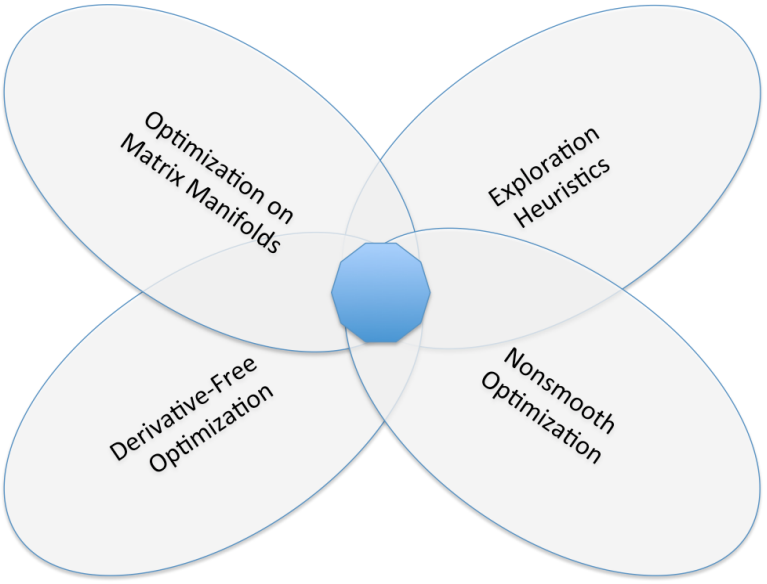


Figure 1.1: The work presented in this thesis lies at the intersection of different optimization fields.

Optimization on matrix manifolds

"Let no one ignorant of geometry enter." - Plato

One of the recurrent themes underlying the studied applications is the concept of optimization on manifolds. This special class of constrained optimization is a recent and vibrant research area (see e.g. [AMS08]). The key idea is to exploit the geometric nature of the constraints using differential geometry concepts. The main resulting advantage, used throughout this thesis, is the opportunity to turn non-linearly constrained optimization problems into unconstrained problems (in a constrained search space). Using differential geometry tools, it is then possible to design algorithms producing sequences of feasible iterates at all times (see Figure 1.2). Another important usage of optimization on manifolds is the opportunity to deal with invariance properties that complicate or weaken convergence analyses, or even adversely affect the actual behaviour of algorithms. By working on quotient manifolds, one can combine equivalent solutions in a single isolated representative element. Recently, the framework of optimization on manifolds has received a lot of attention and has been successfully used in as various contexts as invariant subspace computation (Absil et al. [AMS04], [ASDM04]), gene expression analysis (Journée [Jou09]), geometrical coordination control (Sarlette [Sar09]), higher-order tensors approximation (Ishteva [IAHL11]), linear regression on fixed-rank matrices (Meyer [Mey11]), particle filtering (Rentmeesters et al. [RAD+10]) or matrix completion problems (Boumal and Absil [BA11]).

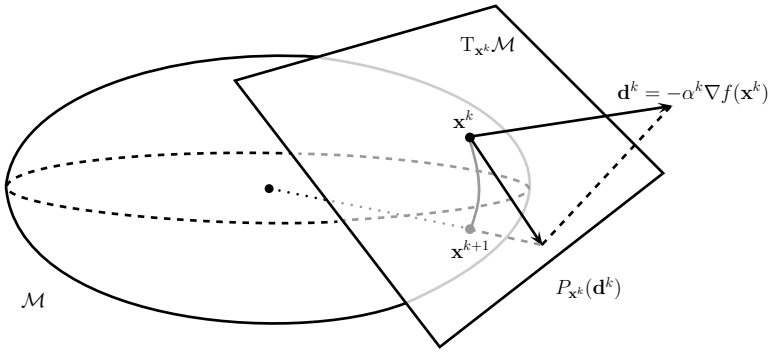


Figure 1.2: Illustration of a steepest-descent step in a Riemannian manifold setting.

Exploration heuristics

"Premature optimization is the root of all evil." - Donald Knuth

Another characteristic of the studied applications is the presence of (possibly many) multiple local minima in the landscape of their objective functions. This makes the task of finding an interesting minimum complex as it implies avoiding to converge prematurely to the first encountered valleys. Numerous optimization heuristics aim at performing this challenging task. The field of optimization heuristics is gaining momentum in the literature and dozens of methods are available nowadays. Often inspired by natural processes like biology (e.g. Artificial Immune System [CC03], Bacterial Foraging Algorithm [DBDA09]), evolution processes (e.g. Genetic Algorithm [Gol89]), social behaviours (e.g. Particle Swarm Optimization [Eng06], Artificial Bee Colony Optimization [KA09]) or physics (e.g. Simulated Annealing [vLA87]), these heuristics are sometimes referred to as metaheuristics. While some of these techniques are designed to present a local search component, they rarely offer interesting convergence properties. However, all these techniques propose more or less sophisticated mechanisms to provide global exploration of the search space. Also, they often lend themselves nicely to hybridization with more efficient local methods. In this context, we apply two of these methods in our work: the Particle Swarm Optimization algorithm (PSO, [KE95]) and the Differential Evolution technique (DE, [SP97]).

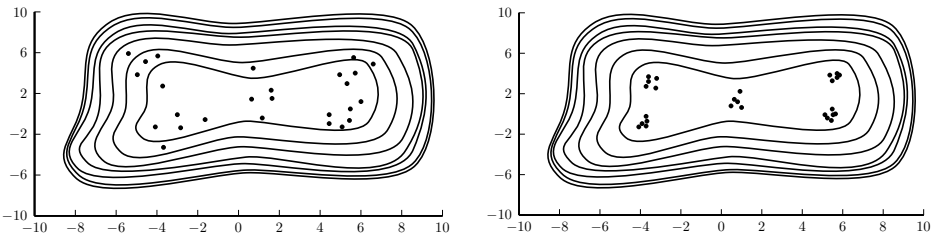


Figure 1.3: Illustration of a population-based exploration heuristic on a 2-dimensional search space.

Derivative-free optimization

"Knowledge is power. Information is liberating. Without it, we are all blind." - Kofi Annan

The extensive amount of information contained in the derivatives of an objective function is the foundation for most standard continuous optimization techniques. However, for many reasons, these derivatives are not always available or reliable in practical situations. Performing optimization while giving up so much information is not a straightforward task, but there exist nevertheless competitive solutions and the field of derivative-free optimization (DFO) is getting more and more attention [CSV09a]. The main characteristic of the DFO techniques is that they produce sequences of iterates in the search space using function evaluations as the only source of information. While these technique were often disregarded in the past by the mathematical community because of the lack of theoretical analysis, new methods with proofs of convergence started to appear in the 1990's (see e.g. [Wri95], [KLT03]), reviving interest for the DFO field. Nowadays, there exist well studied algorithms presenting strong convergence results [CSV09a].

Of particular interest in our work is the class of directional direct search (DS) algorithms, later extended to generalized pattern search (GPS) [Tor97]. In this thesis, we focus on a specific instance of GPS, namely the Mesh Adaptive Direct Search (MADS) [ACD07], presenting strong convergence results while being flexible enough to be hybridized with exploration heuristics.

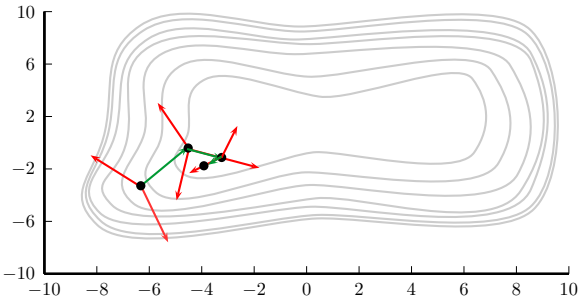


Figure 1.4: Illustration of a direct search algorithm. A sequence of iterates (black dots), the unsuccessful trial directions (red arrows) and successful directions (green arrows) are displayed.

Nonsmooth optimization

"Always take hold of things by the smooth handle." - Thomas Jefferson

While derivative-free optimization is perfectly suitable when the derivatives are unavailable or unreliable, it is often used in practice in situations where this information is available to a large extent. However, as clearly stated by the main contributors to this field: if one can obtain derivatives (even at considerable cost), one should not use derivative-free techniques. In all the applications we study, the objective functions are differentiable almost everywhere. In the case of the ELDP application (Chapter 7), we will produce a tractable expression of the Clarke generalized gradient and resort to elements from Clarke's generalized calculus [Cla75] to perform nonsmooth optimization. We will present a subgradient descent method exploiting the structure of the optimization landscape.

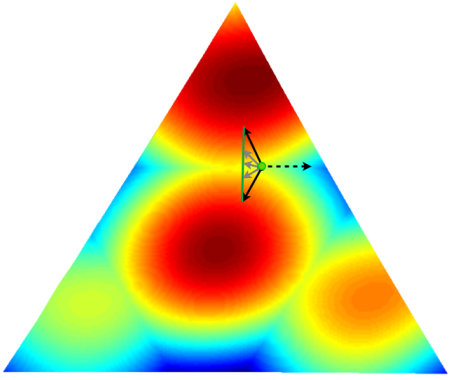


Figure 1.5: Illustration of a subgradient descent step. Some subgradients are illustrated (grey arrows) and the descent direction (dashed arrow) is chosen as the opposite of the shortest vector in the generalized gradient (green line).

While the various optimization domains presented above are all well established, they do not offer complete solutions to the studied applications when applied individually. The work presented in this thesis is mainly an attempt to propose efficient combinations of these concepts, taking an applied engineer point of view.

1.2 Contributions of the thesis

This thesis belongs to a recent and scarcely explored (but see e.g. the work of Dreisigmeyer [Dre07] and Lageman et al. [Lag07]) research theme that aims at developing dedicated methods for the optimization of nonsmooth and multimodal functions on manifolds.

The contributions of this thesis are in a rather applied form, as explained above. We study a series of nonsmooth and/or multimodal optimization problems on matrix manifolds, with a progressive approach in mind. The different challenges presented by these optimization problems are analysed and ad-hoc solutions are designed, with increasing level of sophistication.

We propose specialized algorithms, combining techniques from various optimization fields, to produce adapted solutions to these complex problems. We revisit existing optimization schemes, ranging from smooth optimization techniques to exploration heuristics, along with derivative-free and nonsmooth methods, and adapt them to different manifold settings. Finally, we present how concepts from standard optimization techniques can be combined with ideas coming from the metaheuristics community to design hybrid solutions, providing global exploration of the search space, while ensuring robust local convergence properties.

To summarize, here are the main topics that we contributed to during our research:

1. *Global exploration heuristics on manifolds*
2. *Derivative-free optimization on manifolds*
3. *Nonsmooth optimization on manifolds*
4. *Hybrid exploration-exploitation schemes on manifolds*

Manopt: A Matlab toolbox for optimization on matrix manifolds

The variety of methods studied during our research represent only a small portion of the optimization techniques available nowadays. However, the mechanisms presented in the four applicative chapters to adapt existing methods to manifold settings are often general enough to be

reusable. In this spirit, the development of a Matlab toolbox for optimization on manifolds started a few months ago in our research group. The intent of this toolbox is to allow the development of new methods and/or geometries in a user-friendly way (more information on Manopt is available at `http://www.manopt.org`). By making common geometries and well studied solvers readily available, such a toolbox allows one to push numerical experiments further, while making comparisons and benchmarking much easier than having to manually and individually implement the methods and geometries.

1.3 Organization of the thesis

This thesis is divided into two main parts and is meant to be organized in an application-driven fashion. Optimization being the central theme, a first part (Chapters 2–3) is devoted to introducing generalities about optimization and presents the applications studied during our research. Given the extremely wide scope of optimization in general, the emphasis is put on situating more precisely these applications within this field. The concept of optimization on manifolds is a recurrent topic in this thesis and is therefore also presented in this first part.

The second part (Chapters 4–7) is the core of this thesis. It can be seen as a guided journey through four applications. Starting from a somewhat naive pure derivative-free optimization canvas, more and more insight about the structure and the nature of the problems is used, leading to custom optimization methods for each of the studied applications.

Along the way, we expose systematic ways for adapting existing optimization techniques to manifold settings and discuss general hybridization schemes.

To conclude this work, we present some perspectives for future work, alongside with our conclusions and bibliographic references.

Thesis outline and related publications

Chapter 2: Motivations, material and methods

In this first background chapter, we briefly introduce the key ingredients that will be used through the development of the four practical applications studied during our research. We start from a general overview of optimization concepts and we rapidly restrict this short introduction to the subclasses of interest in this work. We then give a short overview of the applications and show how they motivated us to study different classes of optimization methods.

Chapter 3: Optimization on matrix manifolds

This chapter introduces the notion of optimization on matrix manifolds, which will be thoroughly used in the applications presented in the subsequent chapters. We give an overview of some basic elements of differential geometry and we present the main ingredients of optimization on Riemannian manifolds. The objective of this introductory chapter is to prepare the field for the adaptation of existing optimization methods to manifold settings.

Chapter 4: Minimal volume oriented bounding box

Some of the material of this chapter appeared in the following publication:
[BA10] P.B. Borckmans and P.-A. Absil. Oriented Bounding Box Computation Using Particle Swarm Optimization. *In Proceedings of the 18th European Symposium on Artificial Neural Networks*, ESANN 2010.

In this first applicative chapter, we study the Oriented Bounding Box (OBB) problem. This problem consists in finding the minimal-volume oriented bounding box (OBB) enclosing a given set of points in $\mathbb{R}^3$. Yet simple to state, this problem is computationally challenging, mainly due to the multimodal and nonsmooth characteristics of the associated cost function. We propose two methods based on the Particle Swarm Optimization (PSO) and the Differential Evolution (DE) metaheuristics to provide exploration of the multimodal search space, working on the set of rotation matrices SO(3).

Chapter 5: Higher-order tensor approximation

The material of this chapter is based on the following publication:
[BIA10] P.B. Borckmans, M. Ishteva, and P.-A. Absil. A Modified Particle Swarm Optimization Algorithm for the Best Low Multilinear Rank Approximation of Higher-Order Tensors. *In Proceedings of the 7th international conference on Swarm intelligence*, ANTS10, pages 1323, Berlin, Heidelberg, 2010. Springer-Verlag.

In this second applicative chapter, we are interested in finding the best low multilinear rank approximation of a given tensor. The problem is formulated as an optimization problem over the Grassmann manifold and the objective function presents multiple minima. In order to investigate the landscape of this cost function, we propose an adaptation of the Guaranteed Convergence Particle Swarm Optimization algorithm (GCPSO) to the Cartesian product of Grasmann manifolds.

Chapter 6: Independent component analysis using a range contrast function

This chapter closely corresponds to the following recently-submitted collaborative work:
[BCA13] S. Easter Selvan, Pierre B. Borckmans, Amit Chattopadhyay, and P.-A. Absil. Spherical Mesh Adaptive Direct Search for Separating Quasi-uncorrelated Sources by Range-Based Independent Component Analysis. *Neural Comput.*, 2013, accepted.

This third applicative chapter presents an Independent Component Analysis (ICA) variation, where we extract quasi-uncorrelated sources with finite supports by optimizing a range-based contrast function. This contrast function presents some interesting properties (e.g. the absence of mixing local optima), but is not differentiable everywhere. We therefore design a nonsmooth optimizer on the Oblique manifold. We show that the proposed algorithm ensures convergence to a Clarke stationary point.

Chapter 7: Economic load dispatch with valve-point effect

The material of this chapter is based on the following publication:
[BSBA13] Pierre B. Borckmans, S. Easter Selvan, Nicolas Boumal, and P.-A. Absil. A Riemannian Subgradient Algorithm for Economic Dispatch with Valve-Point Effect. *J. Comput. Applied. Math.*, 2013, accepted.

In this last applicative chapter, we study the economic load dispatch problem (ELDP), a classical problem in the power systems community. It consists in the optimal scheduling of the output of power generating units. Embracing the geometry of the admissible set, the non-differentiable cost function and the multimodal landscape, we propose a Riemannian subgradient descent algorithm to provide fast and robust convergence to local minima. We show how Clarke's calculus can be used to compute deterministic admissible descent directions. Finally, we show that the proposed algorithm can be incorporated in existing heuristic techniques to provide better exploration of the search space.

Part I

Background material

Chapter 2

Motivations, material and methods

Chapter overview
In this first background chapter, we briefly introduce the key ingredients that will be used through the development of the four practical applications studied during our research. Optimization being at the center of this work, we start from a brief overview of this field and we present some generalities and notations. However, since optimization is a very wide research domain, we intentionally restrict this introduction to the subclasses of interest in our work. Once these optimization foundations are set up, we give a short overview of the studied applications. We present the main inherent challenges and show how they motivated us to study, apply and hybridize different classes of optimization algorithms. Given the application-driven spirit of this thesis, the intent is to keep this chapter short while more specific details will be given later in the text, alongside with the development of the applications.

2.1 Optimization generalities

Optimization, also referred to as mathematical programming, is one of the most important fields of applied mathematics. Given an *objective function* (sometimes called cost function, fitness, or quality measure) and a *feasible set* (sometimes called search space or optimization domain), the purpose of an optimization solver is to search for the best admissible solution in the feasible set (*global optimum*) or in a neighbourhood contained in this set (*local optimum*). Typically, the best solution corresponds to the lowest value of the objective function (*minimization problem*). The opposite situation (maximization) is also often encountered and is strictly equivalent.

A general optimization problem can be stated as follows:

$$\min_{x \in \Omega} f(x),$$

where f is the *objective function*, x describes the *decision variables* and Ω is the *feasible set*, possibly defined with the help of constraints.

Definition 2.1. A point x^* is a *global minimum* if $f(x^*) \leq f(x), \forall x \in \Omega$.

A weaker concept is that of locally optimal solutions:

Definition 2.2. A point x^l is a *local minimum* if

$$\exists \epsilon > 0, f(x^l) \leq f(x), \forall x \in B(x^l, \epsilon) \cap \Omega,$$

where $B(x^l, \epsilon)$ is the ball of radius ϵ around x^l.

All globally optimal solutions are also locally optimal, while the opposite is not true for multimodal problems (where several disjoint locally optimal solutions exist). Reaching the global optimum of a given optimization problem is a complicated task and is impossible to guarantee in general. In most cases, one can only hope to find an interesting local minimizer.

Depending on the nature of the problem, the main ingredients x, f and Ω can be expressed in different forms. Most often, the decision variable $\boldsymbol{x}$ is a vector in $\mathbb{R}^n$ and f is a function from $\mathbb{R}^n$ to $\mathbb{R}$. The *feasible set* Ω is then defined as:

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid g_i(\boldsymbol{x}) = 0 \, \forall i \in E, \, g_i(\boldsymbol{x}) \leq 0 \, \forall i \in I\},$$

where $g_i(\boldsymbol{x}) : \Omega \mapsto \mathbb{R}$ play the role of equality constraints for $i \in E$ and inequality constraints for $i \in I$.

However, the decision variables can be different objects. In discrete optimization for instance, x is a vector of integer or binary numbers. In mixed-integer programming, x is a vector combining real-valued with integer entries.

In this work, we focus on continuous optimization. But even in this case, x can be a different object. We will indeed often study optimization problems where the decision variable is a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times m}$. The feasible set Ω is then a subset of the set of $(m \times n)$-matrices. In this case, the constraints are expressed as functions of matrices and so is the objective function:

$$\min_{\boldsymbol{X} \in \Omega} f(\boldsymbol{X}),$$

$$\Omega = \left\{ \boldsymbol{X} \in \mathbb{R}^{n \times m} \mid g_i(\boldsymbol{X}) = 0 \, \forall i \in E, \, g_i(\boldsymbol{X}) \leq 0 \, \forall i \in I \right\}.$$

When the feasible set Ω is a whole vector space, the problem is termed *unconstrained*, while in *constrained optimization* the vector space is restricted by means of equality and/or inequality constraints.

When all the functions (f and g_i) are linear, the optimization problem is called a *linear program*, otherwise it is a *non-linear program*.

2.2 Optimization classes and characteristics

The distinctions about the nature of the decision variables and the objective function presented above are just two of the countless variations for describing an optimization problem. Similar discussions can be made about many aspects of x, f and Ω and Figure 2.1 depicts only a few of these distinctions.

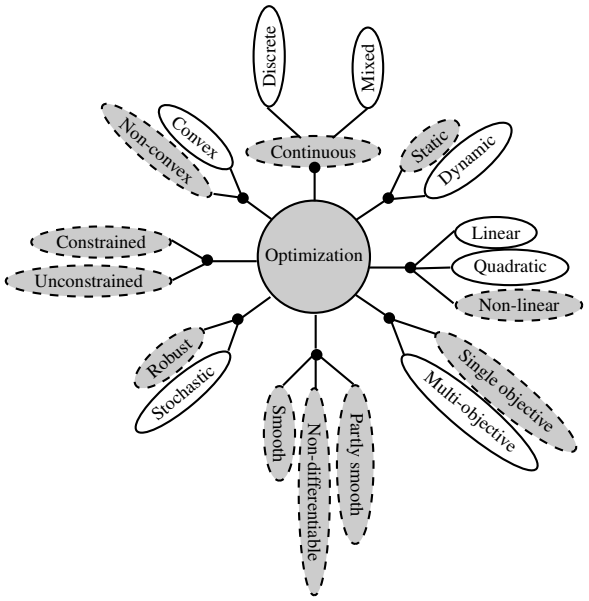


Figure 2.1: A small ensemble of characteristics to distinguish optimization problem classes. Only the features that are greyed out are considered in this work.

Combining the numerous possible variations for these characteristics leads to a huge set of optimization problem classes. Some classes of objective functions are studied in detail in the literature (linear, quadratic, ...) and specific solutions are designed for these problems, using the structure of the problem (e.g. linear programming (LP), quadratic programming (QP), semi-definite programming (SDP), ...). Since it would be naive to try to gather an exhaustive list of existing classes, we now focus on some characteristics that are relevant in the context of our research.

2.2.1 Convexity

Definition 2.3. A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *convex* if:

$$f(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y}),\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n,$$

for all non-negative scalars α and β such that $\alpha + \beta = 1$.

Definition 2.4. A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *concave* if $(-f)$ is convex.

Definition 2.5. A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *affine* if f is convex and concave.

Definition 2.6. An optimization problem is *convex* if:

- the objective function f is convex ;
- the equality constraints $g_i,\ i \in E$, are affine ;
- the inequality constraints $g_i,\ i \in I$, are convex.

Convex optimization is studied intensively in the literature (see classical textbooks, e.g. [BV04]) and powerful solutions exist to solve this type of problems (e.g. interior-point methods [NN87]). The main advantage about convex optimization is that all locally optimal solutions of a convex problem are also globally optimal.

The non-convex case, which is of interest in our work, is often much more difficult to handle. It is sometimes possible and advocated to reformulate or approximate non-convex problems as convex programs. Another possibility, exploited in this thesis, is to relax the demand to obtain the exact best solution of a problem and to only look for local solutions instead. These two approaches are quite different in essence, and both are acceptable depending on the context. This debate is nicely summarized by Boyd (see [BV04], p.10):

"The art in local optimization is in solving the problem (in the weakened sense of finding a locally optimal point), once it is formulated. In convex optimization these are reversed: The art and challenge is in problem formulation; once a problem is formulated as a convex optimization problem, it is relatively straightforward to solve it.

2.2.2 Smoothness - Regularity

The regularity of the cost function f plays a central role in the determination of a suitable solver.

Definition 2.7. A function is said to be $\mathcal{C}^k$ over Ω if all its derivatives exist and are continuous up to order k over Ω.

Definition 2.8. The directional derivative of a multivariate scalar function $f(\boldsymbol{x}) = f(x_1, \ldots, x_n)$ along a direction $\boldsymbol{v}$ is denoted $D_{\boldsymbol{v}} f(\boldsymbol{x})$ and is defined using the following limit:

$$Df(\boldsymbol{x})[\boldsymbol{v}] = \lim_{t \to 0} \frac{f(\boldsymbol{x} + t\boldsymbol{v}) - f(\boldsymbol{x})}{t}.$$

Definition 2.9. The *dot product* of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ in $\mathbb{R}^n$ is defined as follows:

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i.$$

Definition 2.10. The *gradient* of a function f is denoted ∇f and is defined as the unique vector field whose dot product with any vector $\boldsymbol{v}$ at each point $\boldsymbol{x}$ corresponds to the directional derivative of f along $\boldsymbol{v}$:

$$(\nabla f(\boldsymbol{x})) \cdot \boldsymbol{v} = Df(\boldsymbol{x})[\boldsymbol{v}].$$

Using the canonical basis of the Euclidean space $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$, the gradient can be expressed using the partial derivatives as follows:

$$\nabla f = \frac{\partial f}{\partial x_1}\boldsymbol{e}_1 + \cdots + \frac{\partial f}{\partial x_n}\boldsymbol{e}_n = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)^\top.$$

When f is $\mathcal{C}^1$, its gradient exists and is well defined everywhere in the domain of f. If the gradient is sufficiently easy to compute, then first-order optimization techniques such as the steepest descent method (presented in Section 2.4.1) can be applied to obtain linear convergence.

Definition 2.11. The *Hessian* of a $\mathcal{C}^2$ function $f : \mathbb{R}^n \mapsto \mathbb{R}$, denoted $H(f)$, is a square matrix of size $n \times n$ containing the second-order derivatives of f:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

If f is $\mathcal{C}^2$, its Hessian can be computed and higher-order methods such as the Newton method [NN87] can be used. The computational cost of this method can be reduced by approximating the Hessian (Quasi-Newton methods). Other well-known methods in this class are the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method [Sha70] and Broyden's method [Bro65].

When the gradient of f is not available, one can sometimes resort to estimations (e.g. finite differences) to achieve the same result. However, this last option may not suitable if the approximation is too costly.

Definition 2.12. A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is *locally Lipschitz* if, for every bounded subset B of $\mathbb{R}^n$, there exists a real constant $K \geq 0$ such that, $\forall \boldsymbol{x}, \boldsymbol{y} \in B$,

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq K\|\boldsymbol{x} - \boldsymbol{y}\|.$$

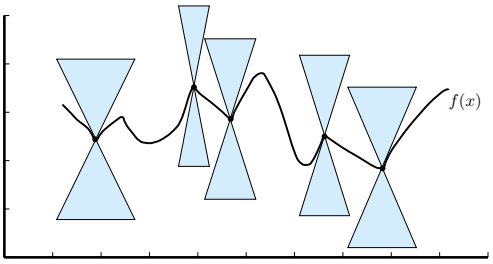


Figure 2.2: Illustration of a Lipschitz function. At each point of the graph of f, a double cone can be drawn such that the graph remains outside of it.

If the function f is not differentiable but still has some smoothness properties (i.e. f is Lipschitz), there exist some derivative-free methods exhibiting convergence properties, such as the Mesh Adaptive Direct Search algorithm (MADS) proposed by Audet et al. [ACD07].

When f is nonsmooth but convex, all hope is not lost since one can still derive interesting properties, notably thanks to the notions of subderivative, subdifferential and subgradient, introduced by Rockafellar in the early 1960's [Roc63].

For a smooth convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$, the following inequality holds:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}),\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n,$$

i.e. the first order approximation of f at $\boldsymbol{x}$ is a global lower bound of f. We now define the concepts of subderivative and subgradient that allow the extension of this result for nonsmooth convex functions.

Let $f : L \mapsto \mathbb{R}$ be a nonsmooth convex function defined over an open interval $L \subseteq \mathbb{R}$.

Definition 2.13. A real number s is a *subderivative* of f at $x \in L$ if

$$f(y) \geq f(x) + s(y - x), \forall y \in L.$$

The set of all such subderivatives of f at a point x for a convex function is a non-empty closed interval (if x belongs to dom$f$) and is called the *subdifferential* of f at x.

These concepts are illustrated in Figure 2.3 and can be generalized to multivariate functions. Let now $f : \Omega \mapsto \mathbb{R}$ be a convex function defined on a convex open set $\Omega \subseteq \mathbb{R}^n$.

Definition 2.14. A vector $\boldsymbol{s}$ is a *subgradient* of f at $\boldsymbol{x} \in \Omega$ if

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{s}^\top(\boldsymbol{y} - \boldsymbol{x}), \forall \boldsymbol{y} \in \Omega.$$

The set of all subgradients at a point $\boldsymbol{x}$ for a convex function is known to be a non-empty convex compact set (again, if $\boldsymbol{x}$ belongs to dom$f$) called the *subdifferential* of f at $\boldsymbol{x}$ and is denoted $\partial f(\boldsymbol{x})$.

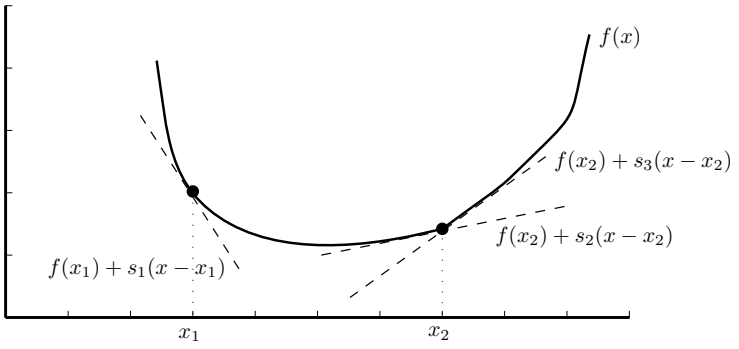


Figure 2.3: Illustration of the concept of subgradient. The function f is differentiable at the point x_1, so the only subgradient is given by the gradient $s_1 = \nabla f(x_1)$. At the point x_2, f is not differentiable, s_2 and s_3 are two subgradients. The subdifferential is the closed interval $[s_2, s_3]$.

Just as the gradient of a smooth function f corresponds to the normal to the graph of f, the subdifferential of a convex nonsmooth function f corresponds to the set of normals to hyperplanes lying underneath the graph of f. At points where f is differentiable, the subdifferential is reduced to a singleton containing only the gradient of f, i.e. $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.

In this thesis, we are mainly interested in solving optimization problems where the objective functions are nonsmooth, locally Lipschitz and non-convex. Thanks to Clarke's generalized calculus framework [Cla75],

the notion of subdifferential can be extended to this class of functions, using the fact that a locally Lipschitz function is differentiable almost everywhere (see e.g. [Ste71]).

Definition 2.15. The *Clarke generalized gradient* $\partial^C f$ of a nonsmooth non-convex locally Lipschitz function f at a point $\boldsymbol{x}$ is defined as the convex hull of limits of gradients at nearby differentiable points:

$$\partial^C f(\boldsymbol{x}) = \operatorname{co}\{\lim \nabla f(\boldsymbol{x} + \boldsymbol{h}_i)\},$$

where $\boldsymbol{h}_i \to 0$ as $i \to \infty$.

The generalized gradient $\partial^C f$ is a nonempty convex compact set. For convex functions, this set reduces to the subdifferential, i.e. $\partial^C f = \partial f$. For smooth functions and at differentiable points of nonsmooth functions, it reduces to the gradient, i.e. $\partial^C f = \{\nabla f\}$.

In general, the sets ∂f and $\partial^C f$ can be hard to obtain in practice. However, for some classes of functions, these sets can be expressed as the convex hull of a finite number of vectors. In particular, this is true if f can be defined as the pointwise maximum of m smooth functions $f_j,\ j = 1, \ldots, m$, which is of interest in our work:

$$f(\boldsymbol{x}) = \max_{j=1,\ldots,m} f_j(\boldsymbol{x}).$$

Defining the set of indices for which the maximum in (2.2.2) is attained:

$$\mathcal{I}_f(\boldsymbol{x}) = \{j \in \{1, \ldots, m\} \mid f(\boldsymbol{x}) = f_j(\boldsymbol{x})\},$$

the generalized gradient can then be expressed as:

$$\partial^C f(\boldsymbol{x}) = \operatorname{co}\{\nabla f_j(\boldsymbol{x}) \mid j \in \mathcal{I}_f(\boldsymbol{x})\},$$

where $\operatorname{co}\{\cdot\}$ denotes the convex hull. In this case, one can design a subgradient descent algorithm as described in Section 2.4.5.

In the general nonsmooth case, one can also resort to other techniques such as derivative-free methods (see Section 2.4.4), heuristics (see section 2.4.3) or stochastic nonsmooth techniques (see e.g. the gradient sampling technique [BLO05]).

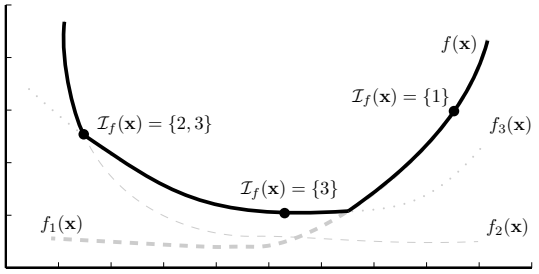


Figure 2.4: Illustration of a piecewise smooth function defined as the maximum of 3 smooth functions. The set of indices $\mathcal{I}_f(\boldsymbol{x})$ for three points are also represented.

2.2.3 Multimodality

In relation with its convexity, one important aspect of a given optimization problem is the presence of multiple local optima within the feasible set Ω. Problems exhibiting this characteristic are called *multimodal* problems. *Global optimization* is the generic name for the category of methods proposing strategies to cope with this challenging property (see e.g. [Wei08]). While there exist many of those techniques, there is no practical method with global convergence guarantee in general. In fact, even for specific classes of problems, it remains extremely hard to design algorithms ensuring convergence to the best minimum.

Applying classical optimization techniques to multimodal problems often yields convergence to minima located in one of the valleys surrounding the initial iterate. Exploration techniques (see Section 2.4.3) aim at providing better investigation of the feasible set Ω while avoiding premature convergence.

2.2.4 Optimality - First order conditions

When the objective function and the constraints of an optimization problem are continuously differentiable, the first-order information can be used to derive optimality conditions. The intuition for deriving those conditions is that for a point $\boldsymbol{x}$ to be a local minimum of f, the value of the first-order approximation (e.g. using Taylor expansion) of f around $\boldsymbol{x}$:

$$f(\boldsymbol{x} + \boldsymbol{\delta}) \approx f(\boldsymbol{x}) + \boldsymbol{\delta}^\top \nabla f(\boldsymbol{x})$$

should be greater than the value of f at $\boldsymbol{x}$, for all directions $\boldsymbol{\delta}$ compatible with the domain Ω:

$$\boldsymbol{\delta}^\top \nabla f(\boldsymbol{x}) \geq 0.$$

For unconstrained problems, all directions are allowed for $\boldsymbol{\delta}$ and this optimality condition is clearly equivalent to the well known condition

$$\nabla f(\boldsymbol{x}) = 0.$$

Given the following optimization problem,

$$\min_{x \in \Omega} f(x),$$

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid g_i(\boldsymbol{x}) = 0 \, \forall i \in E,\, g_i(\boldsymbol{x}) \leq 0 \, \forall i \in I\},$$

where $g_i(\boldsymbol{x}) : \Omega \mapsto \mathbb{R}$ are smooth functions defining equality constraints for $i \in E$ and inequality constraints for $i \in I$, the Karush-Kuhn-Tucker (KKT) optimality conditions extend the necessary optimality condition to constrained problems (see e.g. [NW00]).

Theorem 2.1. *Let the functions f and g_i ($i \in I \cup E$) be continuously differentiable at a point $\boldsymbol{x}^*$. If $\boldsymbol{x}^*$ minimizes $f(\boldsymbol{x})$ locally and satisfies some regularity conditions (see below), then there exist numbers μ_i ($i = 1, \ldots, m$) (called the Lagrange multipliers) such that:*

(a) $g_i(\boldsymbol{x}^*) = 0,\ \forall i \in E$;

(b) $g_i(\boldsymbol{x}^*) \leq 0,\ \mu_i \geq 0,\ \mu_i g_i(\boldsymbol{x}^*) = 0,\ \forall i \in I$;

(c) $\nabla f(\boldsymbol{x^*}) = \sum_{i \in (E \cup I)} \mu_i \nabla g_i(\boldsymbol{x}^*),$

where the values μ_i associated with the constraints $g_i(x),\ i \in I$, are called the Lagrange multipliers.

A point satisfying the KKT conditions is referred to as a (first-order) stationary point.

As mentioned in Theorem 2.1, for a point to satisfy the KKT conditions, some regularity conditions must be respected. Some of the most commonly used are the following:

- **Linearity constraint qualification:** If all the constraints g_i are affine functions, then no condition is needed ;

- **Linear independence constraint qualification (LICQ):** the gradients of the active inequality constraints and the gradients of the equality constraints must be linearly independent at $\boldsymbol{x}^*$;
- **Slater condition:** for a convex problem, there exists a point $\boldsymbol{x}$ such that $g_i(\boldsymbol{x}) = 0,\ i \in E$ and $g_i(\boldsymbol{x}) < 0,\ i \in I$ for all i active in $\boldsymbol{x}^*$.

Note that for convex optimization problems, the KKT conditions are also sufficient for ensuring the global optimality of $\boldsymbol{x}^*$.

When the objective function f is nonsmooth, the necessary condition for a point $\boldsymbol{x}$ to be a local optimum in an unconstrained setting is given by $0 \in \partial f(\boldsymbol{x})$ (or $0 \in \partial^C f(\boldsymbol{x})$ in the non-convex case).

In the constrained case, one can resort to Clarke's calculus to extend the KKT conditions for nonsmooth problems (see [Cla76], Section 3):

Theorem 2.2. *Let the functions f and g_i ($i \in I \cup E$) be locally Lipschitz. If a point $\boldsymbol{x}^*$ minimizes $f(\boldsymbol{x})$ locally, then there exist numbers μ_i ($i = 1, \ldots, m$) such that:*

(a) $g_i(\boldsymbol{x}^*) = 0,\ \forall i \in E$;

(b) $g_i(\boldsymbol{x}^*) \leq 0,\ \mu_i \geq 0,\ \mu_i g_i(\boldsymbol{x}^*) = 0,\ \forall i \in I$;

(c) $\mathbf{0} \in \partial f(\boldsymbol{x^*}) + \sum_{i \in (E \cup I)} \mu_i \partial^C g_i(\boldsymbol{x}^*)$.

A point satisfying these extended KKT conditions is referred to as a *Clarke-stationary point.*

2.2.5 Other considerations

- An objective function can be static or dynamic. In the latter case, the landscape of the function is evolving over time and consecutive calls to the objective function at the same trial point may return different values. Dynamic programming techniques work in this framework [Chi92]. In this work, we only consider static objective functions.
- It is possible to perform optimization for multi-objective functions, i.e. where f is a function from $\mathbb{R}^m$ to $\mathbb{R}^n$. In this case, concepts of Pareto-optimality are used to redefine the notion of optimum [CS03]. This is not considered in this thesis.

2.3 Motivations - Overview of the studied applications

In this section, we briefly present the four applications that will be developed in the subsequent chapters. We show that although these problems look initially quite different from each other, they share a set of common features that motivated us to investigate different fields of optimization methods, presented in the next section.

2.3.1 Overview of the studied applications

Application 1: Oriented Bounding Box (OBB) - Chapter 4

The Oriented Bounding Box (OBB) problem is a classical problem originating from the computational geometry field. This problem consists in finding the minimal-volume oriented bounding box enclosing a given set of points in $\mathbb{R}^3$. This application is notably used in computer vision to perform intersection test between complex objects. Yet simple to state, this problem is challenging, and the state-of-the-art exact method proposed by O'Rourke [O'R85] is computationally too expensive to be applicable on large problem instances.

We study a formulation of the OBB problem as an optimization problem on the rotation group SO(3). Because of the multimodal and nonsmooth characteristics of the associated cost function, global optimization techniques seem to be an appropriate choice. We propose two methods based on the Particle Swarm Optimization (PSO) and the Differential Evolution (DE) metaheuristics to provide global exploration of the multimodal search space.

Application 2: Higher-Order Tensor Approximation (HOTA) - Chapter 5

The multilinear rank of a tensor is one of the possible generalizations for the concept of matrix rank. Finding the best low multilinear rank approximation of a given tensor is a natural extension of the low rank matrix approximation problem. Amongst the many motivations behind these problems, one can cite data compression, dimensionality reduction and independent component analysis.

The problem is formulated as an optimization problem over the Grassmann manifold and the objective function presents multiple min-

ima. In order to investigate the landscape of this cost function, we propose an adaptation of the Guaranteed Convergence Particle Swarm Optimization algorithm (GCPSO) to the Cartesian product of Grassmann manifolds.

Application 3: Range-based Independent Component Analysis (RBICA) - Chapter 6

In this third application, we study a variation of Independent Component Analysis (ICA), where we extract quasi-uncorrelated sources with finite supports by optimizing a range-based contrast function. This contrast function presents some interesting properties (e.g., the absence of mixing local optima), but is not differentiable everywhere. We resort to a derivative-free method, namely the Mesh Adaptive Direct Search (MADS), and we adapt it to the Oblique manifold, to ensure the unit norm constraints inherent to the ICA problem. We show that, while being a nonsmooth optimizer, the proposed algorithm ensures convergence to a Clarke stationary point.

Application 4: Economic Load Dispatch Problem (ELDP) - Chapter 7

In this last application, we study the economic load dispatch problem (ELDP), a classical problem in the power systems community. It consists in the optimal scheduling of the output of power generating units to meet the required load demand subject to unit and system inequality and equality constraints. This optimization problem is challenging on three different levels: the geometry of its admissible set, the non-differentiability of its cost function and the multimodal aspect of its landscape. For this reason, ELDP has received much attention in the past few years and numerous derivative-free techniques have been proposed to tackle its multimodal and non-differentiable characteristics.

Embracing the geometry of the admissible set, the non-differentiable cost function and the multimodal landscape, we propose a Riemannian subgradient descent algorithm to provide fast and robust convergence to local minima. We show how Clarke's calculus can be used to compute deterministic admissible descent directions at all times, by solving a simple low-dimensional quadratic program. Finally, we show that the proposed algorithm can be incorporated in existing heuristic techniques to provide better exploration of the search space.

2.3.2 Main characteristics of the applications

As presented above, the four applications studied in this thesis share at least two common features. They can be expressed as optimization problems on matrix manifolds and they all present a multimodal landscape. Furthermore, applications 1 (OBB), 3 (RBICA) and 4 (ELDP) exhibit a piecewise smooth objective function. The important aspect of application 2 (HOTA) is the presence of multiple minima while the objective function is smooth. Finally, application 4 (ELDP) presents bound constraints on top of its formulation on a manifold.

The main characteristics of these four applications are summarized in Table 2.1.

As explained in the introduction (Section 1.2), the approach taken in this thesis to address these challenges is progressive. We start from a pure heuristic context, using the PSO algorithm to (try to) globally explore the landscape of the first application (OBB). In the second application (HOTA), we reuse the same technique but we include a local scheme (GCPSO) to improve local convergence. For the third application (RBICA), we leave the world of heuristics and we address the problem using the MADS algorithm, a DFO technique with detailed convergence analysis. Finally, for the last application (ELDP), we tend to exploit the structure of the problem as much as possible. We design a subgradient descent algorithm (SD) to cope with the piecewise smooth objective function, we provide mechanisms to deal with bound constraints on a manifold and we incorporate a global heuristic (DE) to enhance the exploration of the search space.

	OBB	**HOTA**	**RBICA**	**ELDP**
Smooth		✓		
Piecewise smooth	✓		✓	✓
Multimodal	✓	✓	✓	✓
Bound constraints				✓
Manifold	SO(3)	$\mathrm{Gr}(p,n)^3$	$\mathcal{OB}(p,n)$	$\mathcal{E}^{n-1}$
Technique	PSO	GCPSO	SMADS	SD+DE
Exploration	✓	✓		✓
Local search		✓	✓	✓

Table 2.1: Summary of challenges and characteristics presented by the four studied applications.

An important remark about our work is that, while the applications are at the center of the developments, their purpose is above all to provide concrete instances of problems with certain characteristics (as detailed in the table above). In other words, although it might be possible to reformulate some of these applications (for instance to make their objective function convex), we intentionally work in their original settings. The overall intent is thus to provide proof-of-concept solutions to deal with nonsmooth and multimodal problems on matrix manifold rather than to provide finely tuned methods achieving the best numerical results for the applications at hand. This also explains why, in the applicative chapters, the emphasis is often put on the development of methods rather than on extensive numerical experiments.

2.4 Material and methods

The short overview of the optimization fields and characteristics developed in Section 2 covers only a small portion of the numerous existing optimization classes. In the rest of this chapter, we specifically address the optimization techniques that we will use in the subsequent chapters and we situate them in their corresponding optimization classes.

2.4.1 Smooth optimization - steepest descent

A very popular method for smooth continuous optimization is the steepest descent algorithm. Given a starting point $\boldsymbol{x}_0 \in \Omega$, the principle is to build a sequence of iterates $\{\boldsymbol{x}_k\}_{k\geq 1}$, attempting to converge to a local minimizer of f. At each iteration, the best direction to move away from the current point $\boldsymbol{x}_k$ is given by the opposite of the gradient $\nabla f(\boldsymbol{x}_k)$. The next iterate $\boldsymbol{x}_{k+1}$ is then computed as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k),$$

where $\alpha_k > 0$ is the step size.

There exist different strategies to determine this step size. Popular choices for computing α_k are given by the Armijo condition and Wolfe conditions [Wol69]. Armijo's rule requires that the trial point $(\boldsymbol{x_k} - \alpha_k \boldsymbol{d}_k)$ achieves sufficient decrease:

$$f(\boldsymbol{x}_k - \alpha_k \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) - \gamma \alpha_k \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{d}_k,$$

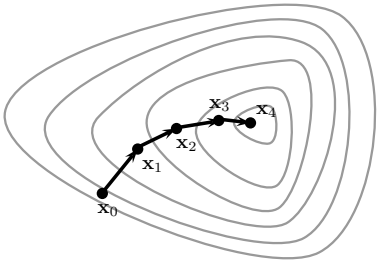


Figure 2.5: Illustration of the steepest descent method.

where $\gamma \in [0, 1]$. Visually, Armijo's rule imposes that the trial point corresponds to a smaller value of f than the linear model of f damped by γ.

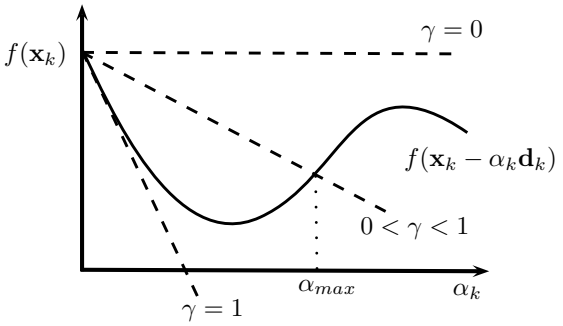


Figure 2.6: Illustration of Armijo's rule for determining the step size α .

The KKT conditions can be used as a stopping criterion, checking for the stationarity of the current iterate.

The steepest descent method is clearly a local method and, as such, its performance in a multimodal context depends highly on the choice of the initial iterate.

2.4.2 Constraints handling - optimization on manifolds

In classical optimization techniques, non-linear inequality constraints are often managed using barrier methods [NW00]. Let us consider a problem with a single inequality constraint $g(\boldsymbol{x})$ and no equality constraints. The common principle behind these techniques is to build a sequence of

unconstrained optimization problems (indexed by k), resembling more and more the initial constrained problem, penalizing the cost function f, e.g. with a log barrier term:

$$\widehat{f}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) + \mu_k \ln(-g(\boldsymbol{x})) & \text{if } g(\boldsymbol{x}) \leq 0, \\ +\infty & \text{otherwise,} \end{cases}$$

where $\mu_k \downarrow 0$ when $k \to \infty$. This technique aims at repelling iterates from the boundaries of the domain Ω to avoid premature convergence to non-stationary points. Similar techniques to handle nonlinear constraints include penalty terms and augmented Lagrangian (see e.g. [NW00], Chapter 17). Each of these subproblems can then be solved using classical unconstrained optimization techniques.

While these approaches can be used to handle inequality constraints, they are not useful for dealing with equality constraints. The framework of optimization on manifolds (see e.g. [AMS08]) on the other hand offers an elegant solution for equality constraints. When the feasible set is a manifold, it is possible, using differential geometry tools, to design algorithms producing sequences of feasible iterates at all times. Another important usage of optimization on manifolds is the opportunity to handle invariance properties of the objective function by working on quotient manifolds.

The concept of optimization on manifolds is present in all the applications studied in this thesis and is therefore discussed in more details in the next chapter.

2.4.3 Exploration - Global optimization heuristics

The presence of (possibly many) multiple local minima in the landscape of an objective function makes the task of finding an interesting minimum complex as it implies avoiding to converge prematurely to the first encountered valleys. Numerous optimization heuristics aim at performing this challenging task. Often inspired by natural processes like biology (e.g. Artificial Immune System [CC03], Bacterial Foraging Algorithm [DBDA09]), evolution processes (e.g. Genetic Algorithm [Gol89]), social behaviours (e.g. Particle Swarm Optimization [Eng06], Artificial Bee Colony Optimization [KA09]) or physics (e.g. Simulated Annealing [vLA87]), these heuristics are sometimes referred to as metaheuristics. The origin of this denomination is twofold. The meta prefix can be explained because of their underlying metaphors. The other reason is

that these methods are designed to be general procedures, applicable to a large variety of problems, and not ad-hoc algorithms tailored for a specific problem. While some of these techniques are designed to present a local search component, they rarely offer interesting convergence properties. This probably explains why the heuristics field is often disregarded by the more theoretical optimization community. However, all these techniques propose more or less sophisticated mechanisms to provide global exploration of the search space. Also, they often lend themselves nicely to hybridization with more robust local methods. In this context, we use two of these methods in our work: the Particle Swarm Optimization algorithm (in applications 1 and 2) and the Differential Evolution technique (in applications 1 and 4).

Particle Swarm Optimization

First introduced in 1995 by Eberhart and Kennedy [KE95], PSO is a stochastic population-based algorithm. Initially presented as a tool to simulate the coordination of individuals in fishes and birds populations, PSO was rapidly adopted by the optimization heuristics community. Particles are points evolving in the search space, following simple rules, mimicking the behaviour of social groups. Points are initialized randomly in $\mathbb{R}^n$ and the driving force of the optimization process is given by the following update equations (iterated over k), for each particle (indexed by i):

$$\begin{aligned} \boldsymbol{v}_i(k+1) &= \underbrace{w(k)\boldsymbol{v}_i(k)}_{\text{inertia}} + \underbrace{c\alpha_i(k)\left(\boldsymbol{y}_i(k)-\boldsymbol{x}_i(k)\right)}_{\text{nostalgia}} + \underbrace{s\beta_i(k)\left(\hat{\boldsymbol{y}}(k)-\boldsymbol{x}_i(k)\right)}_{\text{social}} \\ \boldsymbol{x}_i(k+1) &= \boldsymbol{x}_i(k)+\boldsymbol{v}_i(k+1)\ , \end{aligned}$$

where $\boldsymbol{x}_i$ denotes position, $\boldsymbol{v}_i$ denotes velocity, $\boldsymbol{y}_i$ is the personal best position visited so far, $\hat{\boldsymbol{y}}$ is the global best position of the swarm so far; w is the inertia coefficient (usually dynamic), c and s are adjustable coefficients, and α and β are random components, also called "craziness" in the PSO literature.

As can be seen in Figure 2.7, the behaviour of each particle is dictated by velocity increments composed of three simple components: inertia, cognition (nostalgic behaviour) and social behaviour. At the end of each iteration, $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}$ are updated.

The PSO method has gained a lot of attention since the late 1990's thanks to its interesting exploration properties. Also, this method is often praised for its ability to balance global exploration and local exploitation (through the parameters c and s). Many variations of this

initial implementation are available nowadays, providing specific features such as better local convergence or niching capabilities. For more in-depth information about PSO, the reader is referred to [Eng06].

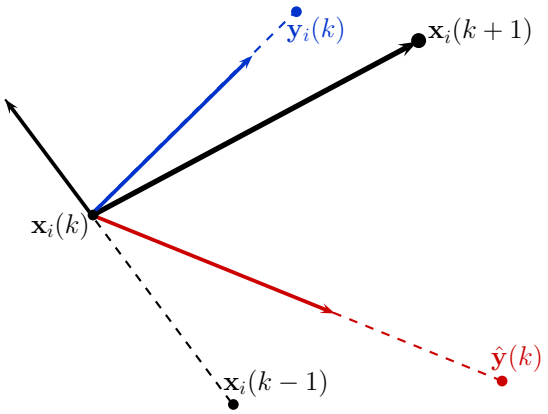


Figure 2.7: Illustration of the update rules for the PSO algorithm. The new iterate $\boldsymbol{x}_i(k+1)$ is computed by combining three contributions: inertia (pursuing previous direction from $\boldsymbol{x}_i(k-1)$), social (leading towards best known solution $\hat{\boldsymbol{y}}(k)$) and nostalgia (heading back to best personal position in the past $\boldsymbol{y}_i(k)$).

An illustration of the swarm evolving in a two-dimensional setup is presented in Figure 2.8.

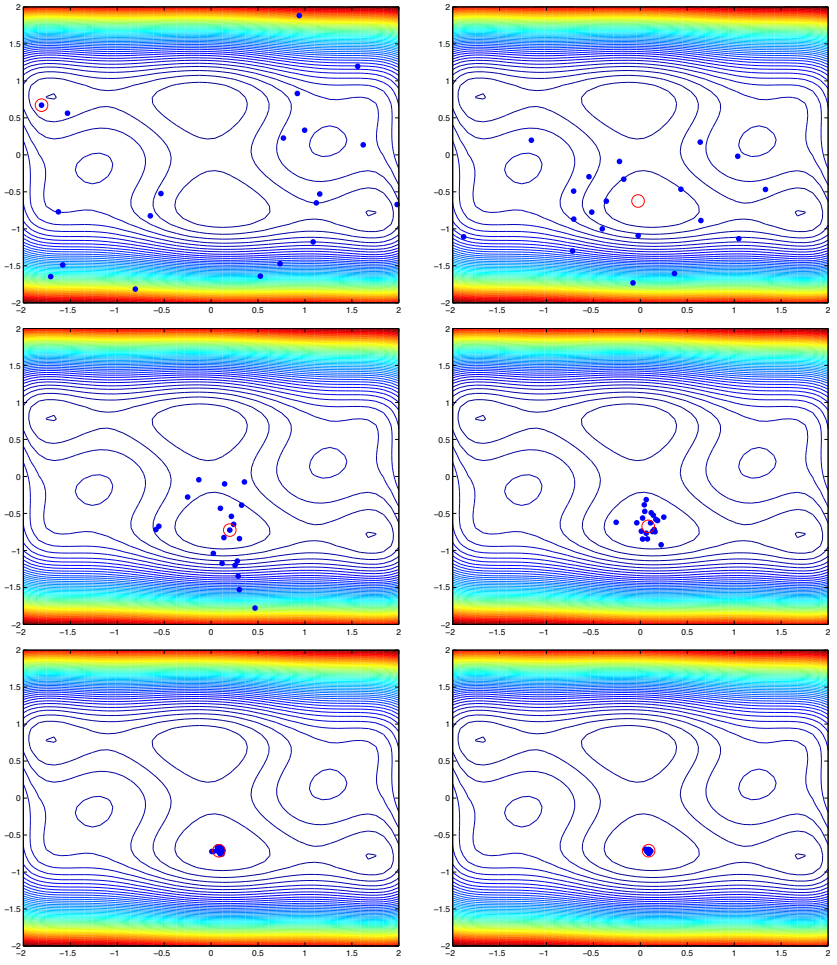


Figure 2.8: Illustration of the PSO algorithm with 20 particles on the six-hump Camel Back function $f(\boldsymbol{x}) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1x_2 + (-4 + 2x_2^2)x_2^2$. The blue dots represent the particles, the red circle denotes the best position discovered so far ($\hat{\boldsymbol{y}}$).

Differential Evolution

The Differential Evolution algorithm (DE) is similar to PSO in that it maintains a population of candidate solutions and creates new candidate solutions by combining existing ones, according to simple rules. Originally due to Storn and Price [SP97] in 1997, this technique has gained popularity and several books have been published on theoretical and practical aspects of using DE in parallel computing, multiobjective optimization and constrained optimization (see e.g. [PSL05], [Cha08], [Feo06]).

The algorithm is initialized with a random population in the n-dimensional search space $(\boldsymbol{x}_i, i = 1, \ldots, m)$. Two parameters are defined:

- The differential weight $F \in [0, 2]$
- The crossover probability $\mathrm{CR} \in [0, 1]$.

In the following, the notation $\boldsymbol{x}_i$ denotes the position of the i-th individual of the population and $\boldsymbol{x}_i^j = \boldsymbol{x}_i^\top \boldsymbol{e}_j$ denotes the j-th component of this position vector ($j \in \{1, \ldots, n\}$). An iteration of the DE algorithm can then be described as follows:

For each individual $\boldsymbol{x}_i$ in the population:

Select three individuals $\boldsymbol{a}_i$, $\boldsymbol{b}_i$ and $\boldsymbol{c}_i$ from the population at random that are distinct from each other as well as from $\boldsymbol{x}_i$;

Pick a random index $R \in \{1, \ldots, n\}$;

Compute the individual's potential new position $\boldsymbol{z}_i$ as follows:

Mutation: Set $\boldsymbol{y}_i = \boldsymbol{a}_i + F(\boldsymbol{b}_i - \boldsymbol{c}_i)$;

Crossover: Compute a random vector $\boldsymbol{r} \sim U(0, 1)^n$;

For each index $j \in \{1, \ldots, n\}$:

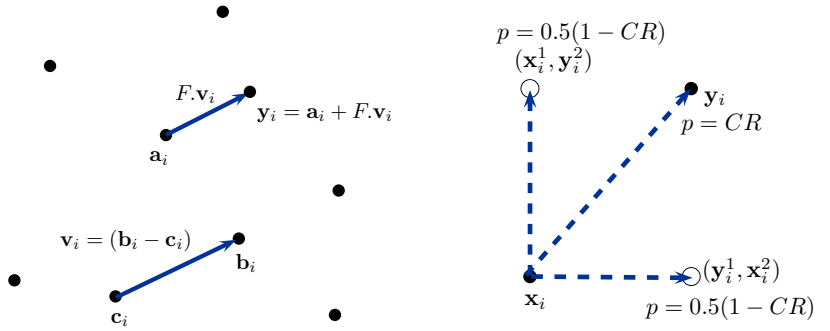
If $(r_j < \mathrm{CR})$ or if $(j = R)$

Set $\boldsymbol{z}_i^j = \boldsymbol{y}_i^j$

Set $\boldsymbol{z}_i^j = \boldsymbol{x}_i^j$ otherwise.

If $f(\boldsymbol{z}_i) < f(\boldsymbol{x}_i)$, set $\boldsymbol{x}_i = \boldsymbol{z}_i$.

Compute the best solution among the population (denoted $\hat{\boldsymbol{x}}$).



(a) Mutation (b) Crossover

Figure 2.9: Illustration of the DE algorithm update equations (mutation and crossover).

As described above and illustrated in Figure 2.9, the DE algorithm consists in two main steps at each iteration and for each individual $\boldsymbol{x}_i$: mutation and crossover. The purpose of the mutation step is to generate a new candidate $\boldsymbol{y}_i$ point that is a combination of three random individuals ($\boldsymbol{x}_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}_c$). This combination is influenced by the differential weight parameter F. Once this candidate is created, the crossover step is used to replace some components j of the current iterate $\boldsymbol{x}_i$ by the corresponding components of the candidate $\boldsymbol{y}_i^j$. The choice of which components are to be replaced is based on the crossover probability CR. Furthermore, one of the components ($j = R$ selected randomly) is always replaced to ensure that at least some modification happens to the current iterate.

The rules used to update the population are rather simple and yet DE is known to perform well on a broad set of problems. The combination of the mutation and crossover steps promotes the investigation of the search space. While this method is not meant to provide efficient local convergence, this ability to explore the landscape globally is often appreciated in practical applications. Therefore, it is often used in hybrid global-local schemes as the exploration tool. We apply this methodology in application 4 (ELDP).

2.4.4 Derivative-free optimization

While global heuristics such as those presented in the previous section present interesting properties for exploring the search space, they are often disregarded because of the lack of convergence analysis in this field.

Derivative-free optimization[1] methods on the other hand are often designed to ensure local convergence under some assumptions. The main characteristic of the DFO techniques is that they produce sequences of iterates in the search space using function evaluations as the only source of information. While these techniques were also often disregarded in the past by the mathematical community, new methods with proofs of convergence started to appear in the 1990's (see e.g. [Wri95], [KLT03]), reviving interest for the DFO field. Nowadays, there exist well studied algorithms presenting strong convergence results [CSV09a].

One popular type of DFO methods are the Trust-region (TR) methods. While this approach was initially designed for the differentiable case (see e.g. [CGT00]), it can be adapted to non-differentiable objective functions. TR methods consist in replacing the objective function locally by a simpler model function (surrogate) in a neighbourhood of the current iterate. In the non-differentiable case, this model is built e.g. using polynomial approximations built on sampling points around the current iterate. At each iteration, the optimization is done on the model function, within a neighbourhood called trust region. The behaviour of the model is then compared to the behaviour of the original cost function and the trust region is adapted consequently. More details on TR methods in the derivative-free context can be found in [CSV09a].

Another important approach in the DFO field is the class of direct search algorithms. A famous example of simplicial direct search is the Nelder-Mead [NM65] method proposed in 1965. Although the NM algorithm is widely used by practitioners, it is not guaranteed to converge to a stationary point [McK98] in its original formulation, and its performance largely depends on the initial configuration of the simplex.

Directional direct search (DS) algorithms, later extended to generalized pattern search (GPS) [Tor97] are also prominent in DFO. The main

[1] Although the heuristics presented in this thesis do not make use of the derivatives, we do not consider them strictly as belonging to the Derivative-Free Optimization techniques category, as they mostly focus on global exploration of the search space. Whenever we mention DFO methods in the sequel, we refer to local optimization methods that do not require derivatives, while presenting convergence guarantees.

idea of these techniques is to inspect a number of trial points around the current iterate, usually in a grid-like setting. In this thesis, we focus on a specific instance of GPS, namely the Mesh Adaptive Direct Search (MADS) [ACD07], presenting strong convergence results while being flexible enough to be hybridized with exploration heuristics.

Mesh Adaptive Direct Search (MADS)

Generalized Pattern Search methods (GPS) are a well known class of derivative-free optimizers. These algorithms address the minimization of nonsmooth functions and are simple to implement, but they only provide limited convergence guarantees, especially for constrained optimization problems. Audet *et al.* proposed an extension of the GPS class, called Mesh Adaptive Direct Search (MADS) [ACD07], allowing local exploration using a dense set of direction in the constrained optimization space. This technique allows the treatment of constraints using the extreme barrier technique (putting the objective function to infinity for points outside the feasible set), while ensuring convergence to a Clarke stationary point under certain assumptions [ACD07] (mainly the Lipschitz continuity of the cost function).

MADS is an iterative algorithm generating a finite number of trial points at each iteration k, using some sets of directions around the current iterate $\boldsymbol{x}_k$. The current incumbent value $f(\boldsymbol{x}_k)$ is compared to the objective function at the candidates (only the ones that are feasible) and is updated when improvement is achieved (without any sufficient decrease requirement). The trial points must belong to a specific mesh, denoted by M_k, constructed using n_D directions in $\mathbb{R}^n$, stored columnwise in a matrix $\boldsymbol{D}$, and scaled by the mesh size parameter $\Delta_k^m \in \mathbb{R}_+$:

$$M_k = \{\boldsymbol{x}_k + \Delta_k^m \boldsymbol{D}\boldsymbol{z} : \boldsymbol{z} \in \mathbb{N}^{n_D}\}\,.$$

Note that in order to ensure the MADS convergence properties, the n_D columns of $\boldsymbol{D}$ must form a positive spanning set of $\mathbb{R}^n$, i.e. any point in $\mathbb{R}^n$ can be obtained as linear combination of the columns of $\boldsymbol{D}$, using only positive coefficients. In practice, this mesh is never built, but the iterates are constructed so as to belong to it.

Each iteration of the MADS algorithm is composed of two steps, namely the *search step* and the *poll step*. The search step is optional and allows the exploration of the search space. Any heuristic can be used to sample candidates, as long as they lie on the mesh M_k. This step is

not involved in the convergence proofs and is typically used to enhance global exploration of the search space. The poll step on the other hand is mandatory, and is used whenever the search step is not used or fails to produce an improved iterate. The poll step is the one that leads to convergence results. To do so, it only allows candidate points that lie on a subset P_k of the mesh M_k, called the *poll frame*:

$$P_k = \{\boldsymbol{x}_k + \Delta_k^m \boldsymbol{d} : \boldsymbol{d} \in \boldsymbol{D}_k\} \subset M_k, \tag{2.1}$$

where $\boldsymbol{D}_k$ is a column-wise concatenation of n_P directions that are nonnegative integer linear combinations of the directions in $\boldsymbol{D}$ ($\boldsymbol{D}_k = \boldsymbol{D}\boldsymbol{U}_k, \boldsymbol{U}_k \in \mathbb{N}^{n_D \times n_P}$) and that form a positive spanning set of $\mathbb{R}^n$. Note that in (2.1), the notation $\boldsymbol{d} \in \boldsymbol{D}_k$ is slightly abused, meaning that $\boldsymbol{d}$ is chosen as one of the columns of $\boldsymbol{D}_k$.

The poll points $\in P_k$ must respect the following condition: the distance from the frame center $\boldsymbol{x}_k$ to a poll point $\boldsymbol{x}_k + \Delta_k^m \boldsymbol{d}$ is bounded by a constant times the poll size parameter: $\Delta_k^m \|\boldsymbol{d}\| \leq \Delta_k^p \max\{\|\boldsymbol{d}'\| : \boldsymbol{d}' \in \boldsymbol{D}\}$. The mesh size parameter Δ_k^m is updated at the end of each iteration, using the following rule, where τ_k and θ_k are positive rational numbers with $\theta_k > 1$:

$$\Delta_{k+1}^m = \begin{cases} \tau_k \Delta_k^m & \text{if the iteration was successful} \\ \frac{1}{\theta_k} \Delta_k^m & \text{otherwise.} \end{cases} \tag{2.2}$$

The poll size parameter Δ_k^p is then chosen so that $\Delta_k^m \leq \Delta_k^p$.

The MADS algorithm can be described as follows:

Initialization:

- Let $\Omega \subset \mathbb{R}^n$ be the feasible set ;
- Initialize $\boldsymbol{x}_0 \in \Omega$ randomly ;
- Set $f_0 = f(\boldsymbol{x}_0)$;
- Let $\epsilon > 0$ and $k = 0$

Iteration k:

- SEARCH STEP: if iteration $(k-1)$ was successful, perform optional search step:
  - Sample a finite set of trial points $\boldsymbol{s}_i, i = 1, \ldots, n_S$ on M_k (using any heuristic);
  - Evaluate f at each $\boldsymbol{s}_i$ and compare with f_k.
- POLL STEP: if iteration $(k-1)$ was unsuccessful or if search step is ignored, perform poll step:
  - Obtain the set of points $\boldsymbol{p}_i, i = 1, \ldots, n_P$ on the poll frame P_k ;
  - Evaluate f at each $\boldsymbol{p}_i$ and compare with f_k.
- PARAMETER UPDATE:
  - If improvement is reached
    update $\boldsymbol{x}_{k+1}$ to the improved candidate ;
    set $f_{k+1} = f(\boldsymbol{x}_{k+1})$;
    declare iteration successful ;
  - Update Δ_k^m following (2.2) and set $\Delta_k^p \geq \Delta_k^m$;
  - Increment k.

Termination: repeat until $\Delta_k^p \leq \epsilon$

The crucial distinction between classical GPS methods and MADS is that if Δ_k^m goes to zero more rapidly than Δ_k^p, then the directions in $\boldsymbol{D}^k$ used to define the poll frame may be selected in a way so that they are not confined to a finite set. Note that GPS methods can be seen as a special case of MADS, in which Δ_k^m and Δ_k^p are identical: a single parameter plays the role of the mesh and poll size parameters, and therefore, the number of positive spanning sets that can be formed by subsets of D is constant over all iterations, as depicted in Figure 2.10. The introduction of the poll size parameter Δ_k^p in the MADS algorithm allows to select poll directions from a much larger set $\boldsymbol{D}^k$, as illustrated in Figure 2.11.

MADS is not properly speaking a method but rather a general canvas offering some local convergence guarantees that are discussed in

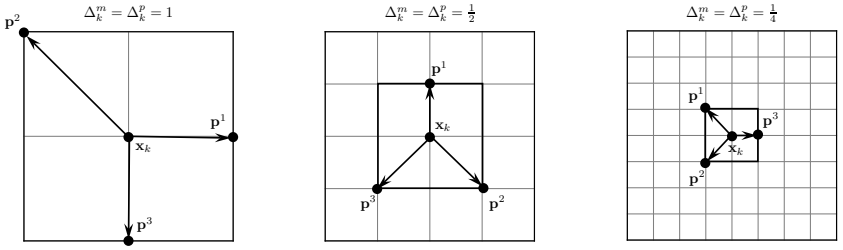


Figure 2.10: Illustration of the GPS algorithm as a special case of the MADS algorithm. The set of directions from which the poll frame $P_k = \{\boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3\}$ can be constructed is always composed of 8 elements, independently of the mesh size parameter Δ_k^m.

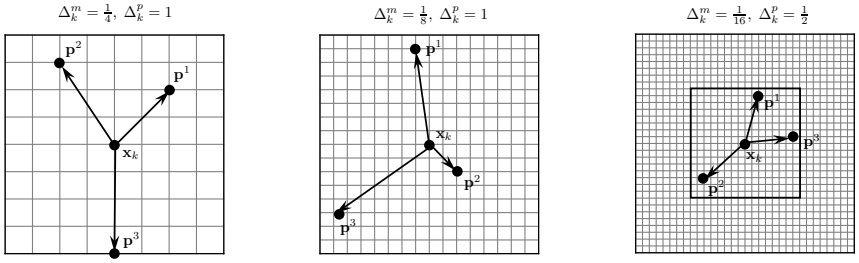


Figure 2.11: Illustration of the MADS algorithm poll step. The set of directions from which the poll frame $P_k = \{\boldsymbol{p}^1, \boldsymbol{p}^2, \boldsymbol{p}^3\}$ can be constructed is larger than in classical GPS algorithms, as the mesh size parameter Δ_k^m goes to zero faster than the poll size parameter Δ_k^p.

[ACD07]. The difficulty is to provide practical implementations of MADS, respecting some required conditions on the set of directions $\boldsymbol{D}_k$ involved in the proofs. Audet *et al.* proposed two schemes for the construction of the directions $\boldsymbol{D}$ and $\boldsymbol{D}_k$, leading to two versions of MADS called LTMADS [ACD07] (involving stochastic directions) and ORTHOMADS [AADD09] (with deterministic directions).

2.4.5 Nonsmooth optimization

As already mentioned before, obtaining the generalized gradient for arbitrary functions is often impossible in practice. Some techniques in nonsmooth optimization try to approximate the generalized gradient. In the gradient sampling method proposed by Overton et al. [BLO05]

for instance, at each iteration, a set of sample points are selected randomly in a neighbourhood around the current iterate. The gradients at these points are evaluated, and the generalized gradient at the current iterate is approximated by the convex hull of these gradients. The size of the neighbourhood is then adapted depending on the norm of the obtained descent direction.

In this thesis, we mainly consider objective functions that are (simple smooth expressions of) the pointwise maximum of smooth functions. The framework presented in this Section can then be used to obtain the generalized gradient of the function at every point in the feasible set and stochastic methods such as the gradient sampling technique are thus not necessary.

Subgradient descent for piecewise smooth functions

In application 4 (ELDP), we propose a subgradient algorithm for piecewise smooth functions. We recall here the framework presented in Section 2.2.2. We consider a piecewise smooth function f defined as the pointwise maximum of m smooth functions:

$$f(\boldsymbol{x}) = \max_{j=1,\dots,m} f_j(\boldsymbol{x}).$$

The set of indices for which the maximum is attained is denoted $\mathcal{I}_f(\boldsymbol{x})$:

$$\mathcal{I}_f(\boldsymbol{x}) = \{j \in \{1,\dots,m\} \mid f(\boldsymbol{x}) = f_j(\boldsymbol{x})\},$$

and the generalized gradient can then be expressed as:

$$\partial^C f(\boldsymbol{x}) = \operatorname{co}\{\nabla f_j(\boldsymbol{x}) \mid j \in \mathcal{I}_f(\boldsymbol{x})\}.$$

Using this framework, the subgradient descent step for the iteration k using a step size α^k is given by

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha^k \boldsymbol{d}^k,$$

where the descent direction $\boldsymbol{d}^k$ is given by the shortest vector in $\partial f(\boldsymbol{x}^k)$, which can be obtained by solving the following quadratic programming problem:

$$\min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| \sum_{j \in \mathcal{I}_f(\boldsymbol{x}^k)} \lambda_j \nabla f_j(\boldsymbol{x}^k) \right\|^2.$$

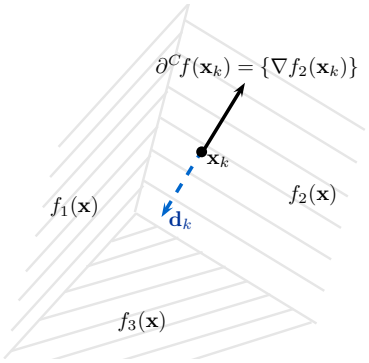


Figure 2.12: Illustration of the subgradient descent algorithm. At a differentiable point $\boldsymbol{x}_k$, the generalized gradient is simply given by the gradient: $\partial^C f(\boldsymbol{x}_k) = \{\nabla f(\boldsymbol{x}_k)\}$. The descent direction is then chosen as the opposite of the gradient.

In practice, when the set $\partial^C f(\boldsymbol{x}^k)$ is composed of a single element $\nabla f_j(\boldsymbol{x}^k)$, the subproblem becomes trivial as the direction is simply given by $\boldsymbol{d}^k = \nabla f_j(\boldsymbol{x}^k)$ (see Figure 2.12). When $\partial^C f(\boldsymbol{x}^k)$ contains at least two elements, solving the subproblem allows to compute a descent direction even though the function is not differentiable. This procedure can be visualized in Figure 2.13, where the illustrative function is defined as the pointwise maximum of three planes.

Finally, the extended KKT conditions can be used as a stopping criterion to check for Clarke-stationarity. In the unconstrained case, the condition is simply given by:

$$0 \in \partial^C f(\boldsymbol{x}_k),$$

as illustrated in Figure 2.14. Intuitively, when the zero vector belongs to the generalized gradient at a given point, all directions around this point are ascent directions and therefore no descent can be achieved.

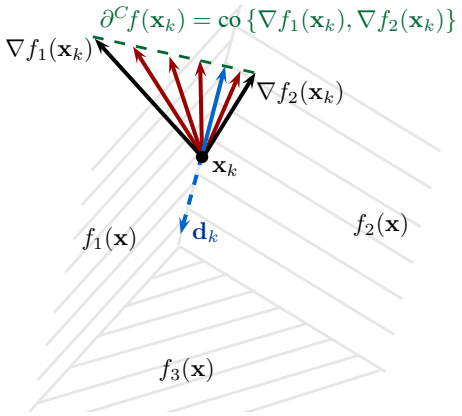


Figure 2.13: Illustration of the subgradient descent algorithm subproblem to determine a descent direction. Some subgradients (red arrows), the generalized gradient (dashed green line) and the descent direction (dashed blue arrow) are represented. The descent direction is obtained as the opposite of the shortest vector in $\partial^C f(\boldsymbol{x}_k)$ (blue arrow).

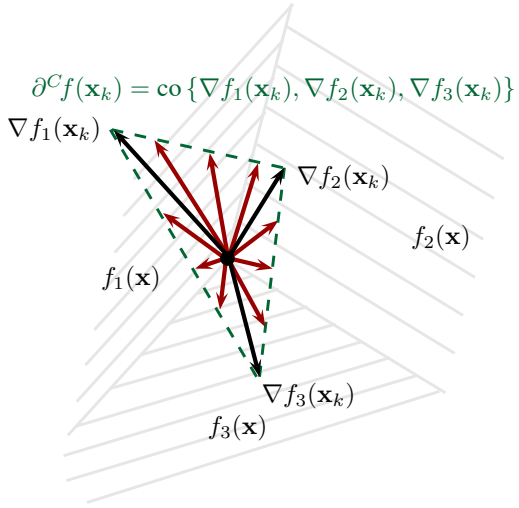


Figure 2.14: Illustration of a Clarke stationary point for a piecewise smooth (linear) function. Some subgradients (red arrows) and the generalized gradient (area inside the dashed green line) are represented. The zero vector belongs to $\partial^C f(\boldsymbol{x}_k)$.

Chapter 3

Optimization on Matrix Manifolds

Chapter overview
This chapter introduces the notion of optimization on matrix manifolds, which will be thoroughly used in the applications presented in the subsequent chapters. We give an overview of some basic elements of differential geometry and we present the main ingredients of optimization on Riemannian manifolds. The objective of this introductory chapter is to prepare the field for the adaptation of existing optimization methods to manifold settings.
After providing some motivation for using this optimization framework, we introduce the concepts of embedded and quotient manifolds. We define the notions of tangent space, geodesic, retraction, alongside with the concepts of exponential map, logarithmic map and parallel transport. Furthermore, we present general schemes for adapting line-search and iterative methods to matrix manifolds in a systematic way.
While the intent of this chapter is to present a general introduction to matrix manifolds, it is clearly not meant to provide a complete reference on this topic and the emphasis is put on giving a working intuition. For a more in-depth coverage of this optimization field, we refer to [AMS08] and [Lee03].

3.1 Motivations

The applications presented in this thesis all share a common feature: their associated constraint sets have a particularly rich geometric structure. More precisely, these constraint sets are matrix manifolds: they are manifolds in the sense of differential geometry and there is a natural matrix representation of their elements. Working in the framework of optimization on matrix manifolds allows one to deal efficiently with these constraints.

The framework of optimization on manifolds offers an elegant solution for dealing with various non-linear equality constraints. The key idea is to embed the constraints into the search space, allowing one to solve an unconstrained optimization problem in a constrained search space. Using differential geometry tools, it is then possible to design algorithms producing sequences of feasible iterates at all times. Another important usage of optimization on manifolds is the opportunity to deal with invariance properties of a cost function that might complicate or weaken convergence analyses, or even adversely affect the actual behaviour of algorithms, by working on quotient manifolds.

The concept of optimization on Riemannian manifolds is not new. Back in 1972, Luenberger already presented this idea in [Lue72]. A few years later, in the control systems community, Brockett worked on differential equations with solutions evolving on a manifold [Bro72]. The monograph by Helmke and Moore in 1994 [HM94] is a major contribution in the field and shows how computational problems can be formulated as optimization algorithms on manifolds. More recently, the book of Absil et al. [AMS08] presents state of the art algorithmic developments and addresses numerical issues related to optimizing smooth functions on a manifold.

In the last decade, the framework of optimization on manifolds has received a lot of attention and has been successfully used in as various contexts as invariant subspace computation ([AMS04], [ASDM04]), gene expression analysis ([Jou09]), geometrical coordination control ([Sar09]), higher-order tensors approximation ([IAHL11]), linear regression on fixed-rank matrices ([Mey11]), particle filtering ([RAD$^+$10]) or matrix completion problems ([BA11]).

An ongoing research theme, notably developed in the work of Dreisigmeyer [Dre07], Lageman et al. [Lag07] and in this thesis, is to extend the development of optimization techniques on manifolds for nonsmooth

functions.

3.2 Matrix manifolds

Defining the notion of manifold in its full generality requires sophisticated topological notions, involving coordinate charts and atlases (see e.g. [AMS08], Chapter 3, pp.18–20). However, for the purpose of our work, it is enough to consider the following simplified definition:

> A *manifold* is a set covered with coordinate patches
> that overlap smoothly.

Manifolds generalize the notion of smooth curves and surfaces. When the set is a subset of another manifold (in particular, of a Euclidean space), then it admits at most one natural manifold structure. Endowed with this structure, the set is termed an *(embedded) submanifold* and the other manifold is termed the *embedding space*. Similarly, when the set is a quotient of another manifold, then it admits at most one natural manifold structure, with which the set is termed a *quotient manifold*, and the other manifold is termed the *total space*.

Intuitively, a manifold can be seen as a smooth surface with a possibly complex shape. As depicted in Figure 3.1, it can always be covered with patches $(\mathcal{U}_1, \mathcal{U}_2, \dots)$ whose union cover the whole manifold. There must exist a collection (atlas) of one-to-one mappings $(\phi_1, \phi_2, \dots)$ between each patch and an open set of the Euclidean space $(\phi_1(\mathcal{U}_1), \phi_2(\mathcal{U}_2), \dots)$, and these mappings must be compatible if some patches overlap. The dimension of the manifold is given by the dimension of these open sets. It is quite impractical to use these mappings in numerical implementations as this involves keeping track of an (possibly intricate) atlas. However, if a manifold presents a matrix structure, one can advantageously resort to matrix algebra to provide natural development of differential geometry. This is the framework used in this thesis and therefore, we will often say manifold to design matrix manifolds in the sequel.

Since a manifold is endowed with a differentiable structure, it is possible to define tangent vectors and to extend classical concepts of differentiation. One can then compute directional derivatives, gradients or Hessians and consequently perform optimization on manifolds. If a differentiable manifolds has an associated inner product in each tangent space, it is called a Riemannian manifold.

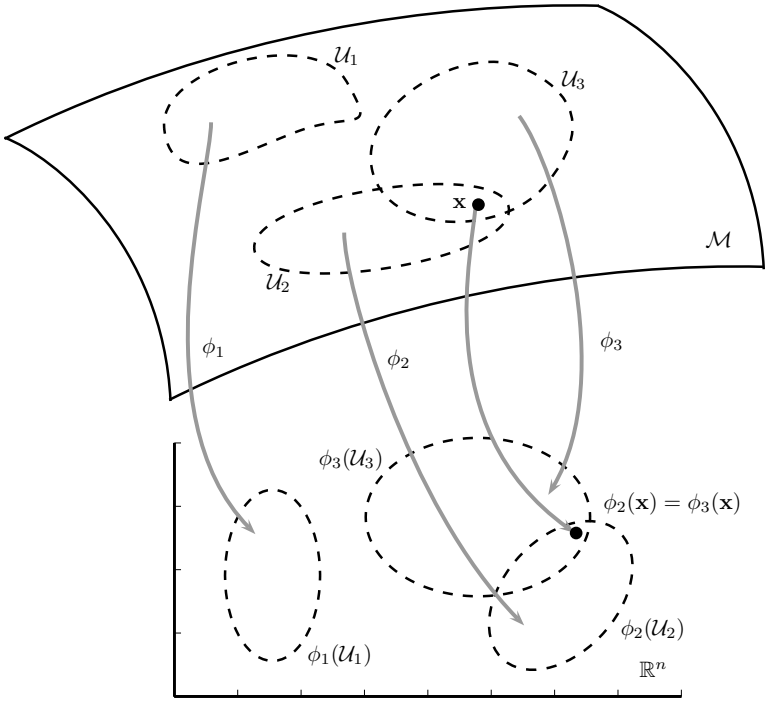


Figure 3.1: Illustration of a smooth manifold $\mathcal{M}$ embedded in $\mathbb{R}^n$. A few patches $\mathcal{U}_i$ are illustrated alongside with the corresponding mappings ϕ_i to regions of $\mathbb{R}^n$.

In this thesis, we will work with two types of manifolds: embedded manifolds and quotient manifolds. We now briefly present these concepts and we refer to [AMS08] and [Lee03] for more in-depth definitions.

3.2.1 Embedded submanifolds

Embedded submanifolds can be viewed as a generalization of the notion of embedded surface in $\mathbb{R}^n$. All the embedded submanifolds considered in this work are defined by explicit algebraic constraints in the matrix space $\mathbb{R}^{m\times n}$. A subset of a manifold is an embedded submanifold if it is endowed with a differential structure that is compatible with the embedding space.

One of the most straightforward examples of embedded manifold is the *unit hypersphere* $S^{n-1} = \left\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^\top \boldsymbol{x} = 1\right\}$. The points satisfying this definition correspond indeed to the surface of a hypersphere of radius 1, embedded in the Euclidean space of dimension n.

In the first application (chapter 4), we work on the *special orthogonal group* $\mathrm{SO}(n) = \left\{\boldsymbol{X} \in \mathbb{R}^{n\times n} \mid \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}_n,\ \det(\boldsymbol{X}) = 1\right\}$. Each point on this manifold corresponds to a rotation matrix.

Another well known embedded manifold that is used in this thesis is the *Stiefel manifold*, $\mathrm{St}(p, n) = \left\{\boldsymbol{X} \in \mathbb{R}^{n\times p} \mid \boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{I}_p\right\}$, i.e. the set of all column-wise orthonormal $(n \times p)$-matrices. Note that the unit hypersphere S^{n-1} corresponds to $\mathrm{St}(1, n)$.

The *oblique manifold* $\mathcal{OB}(p, n)$, used in chapter 6, can be seen as the Cartesian product of p unit spheres. It is defined as

$$\mathcal{OB}(p, n) = \left\{\boldsymbol{X} \in \mathbb{R}^{n\times p} \mid \mathrm{ddiag}(\boldsymbol{X}^\top \boldsymbol{X}) = \boldsymbol{I}_p\right\},$$

where $\mathrm{ddiag}(\cdot)$ is the matrix containing the diagonal of its matrix argument on the diagonal and zeros outside the diagonal.

Finally, in chapter 7, we will define the *Ellipsoid manifold* $\mathcal{E}^{n-1} := \{\boldsymbol{x} \in \mathbb{R}^n \ :\ \boldsymbol{x}^\top \boldsymbol{B}^{\mathrm{ell}} \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b}^{\mathrm{ell}} + c^{\mathrm{ell}} = 0\}$, where $\boldsymbol{B}^{\mathrm{ell}}$ is symmetric positive definite. This manifold can be obtained by an affine transformation of the hypersphere S^{n-1}. The coefficients $\boldsymbol{B}^{\mathrm{ell}}$, $\boldsymbol{b}^{\mathrm{ell}}$ and c^{ell} describe this transformation.

3.2.2 Quotient manifolds

The concept of quotient manifolds is a bit more abstract than embedded manifolds. They are mostly useful when the solutions of an optimization

problem are not isolated in the search space, which is often the source of issues for numerical methods.

In a quotient matrix manifold, each point is defined as an equivalence class of matrices, which allows one to deal with a single representative of equivalent solutions. However, since these equivalence classes are abstract objects and cannot be explicitly encoded in numerical models, algorithms on quotient manifolds usually work with representatives of these equivalence classes. The differential geometry tools needed to perform optimization can then be defined for these representatives, as long as the corresponding definitions do not depend on particular choices for these representatives (see Section 3.3.2).

Two typical examples of quotient matrix manifolds are the real projective space (the set of all directions in $\mathbb{R}^n$, i.e. the straight lines going through the origin) and the Grassmann manifold $\mathrm{Gr}(p, n)$ (the set of all p-dimensional subspaces of $\mathbb{R}^n$). In the real projective space, coinciding with $\mathrm{Gr}(1, n)$, the equivalence classes are defined by parallel vectors since any scaling of a vector in $\mathbb{R}^n$ does not affect its direction. In the Grassmann manifold, each subspace is defined by a p-dimensional orthogonal frame, which can be arbitrarily rotated, leading to equivalence classes.

3.3 Differential geometry tools

3.3.1 Definitions and concepts

Differential geometry is the mathematical theory that provides the tools to efficiently deal with manifolds. In the following, we define some concepts from this field that will be used through our applications.

Let $\mathcal{M}$ be a differentiable manifold and $\boldsymbol{p}$ a point of $\mathcal{M}$.

Definition 3.1. The *tangent space* to a submanifold $\mathcal{M}$ of $\mathbb{R}^n$ at a point $\boldsymbol{p} \in \mathcal{M}$, denoted by $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, is the vector subspace of $\mathbb{R}^n$ defined as:

$$\mathrm{T}_{\boldsymbol{p}}\mathcal{M} = \{\boldsymbol{\xi} \in \mathbb{R}^n \mid \exists c : \mathbb{R} \to \mathcal{M} \text{ with } c(0) = \boldsymbol{p},\ c'(0) = \boldsymbol{\xi}\},$$

where $c'(0)$ is the usual derivative at 0 of the curve c (assuming it exists).

From this definition, we see that $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is the set of vectors that are tangent to the manifold at $\boldsymbol{p}$. Geometrically, this notion coincides with the concept of tangent plane to a smooth surface, as depicted in Figure 3.2.

Definition 3.2. The tangent bundle $\mathrm{T}\mathcal{M}$ is the collection of the tangent spaces at all $\boldsymbol{p} \in \mathcal{M}$.

Each tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is a vector space and as such can be endowed with an inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{p}}$. A classical way of doing this is by restricting the canonical inner product of $\mathbb{R}^n$ to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, i.e.

$$\langle ., . \rangle_{\boldsymbol{p}} : \mathrm{T}_{\boldsymbol{p}}\mathcal{M} \times \mathrm{T}_{\boldsymbol{p}}\mathcal{M} \to \mathbb{R} : (\boldsymbol{\xi}, \boldsymbol{\zeta}) \mapsto \langle \boldsymbol{\xi}, \boldsymbol{\zeta} \rangle_{\boldsymbol{p}} = \langle \boldsymbol{\xi}, \boldsymbol{\zeta} \rangle = \mathrm{trace}\left(\boldsymbol{\xi}^{\top}\boldsymbol{\zeta}\right).$$

We then say that $\mathcal{M}$ is a *Riemannian submanifold* of $\mathbb{R}^n$. The inner product induces a notion of norm: $\|\boldsymbol{\xi}\|_{\boldsymbol{p}} = \langle \boldsymbol{\xi}, \boldsymbol{\xi} \rangle_{\boldsymbol{p}}^{1/2}$.

An *affine connection* is a geometric object on a smooth manifold which connects nearby tangent spaces, and so permits tangent vector fields to be differentiated as if they were functions on the manifold with values in a fixed vector space.

Definition 3.3. The *normal space* to a manifold $\mathcal{M}$ at a point $\boldsymbol{p} \in \mathcal{M}$, denoted as $\mathrm{N}_{\boldsymbol{p}}\mathcal{M}$, is the orthogonal complement of $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$:

$$\mathrm{N}_{\boldsymbol{p}}\mathcal{M} = (\mathrm{T}_{\boldsymbol{p}}\mathcal{M})^{\perp}.$$

One can then compute the *projection* $P_{\boldsymbol{p}}(\boldsymbol{v})$ of a vector $\boldsymbol{v} \in \mathbb{R}^n$ onto $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ by removing the normal component of $\boldsymbol{v}$.

Definition 3.4. A *retraction* on $\mathcal{M}$ is a smooth mapping R from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ that satisfies $R(0_{\boldsymbol{p}}) = \boldsymbol{p}$ for all $\boldsymbol{p}$ (where $0_{\boldsymbol{p}}$ denotes the origin of $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$) and $\frac{\mathrm{d}}{\mathrm{d}t} R(t\boldsymbol{\xi}_{\boldsymbol{p}})\big|_{t=0} = \boldsymbol{\xi}_{\boldsymbol{p}}$ for all $\boldsymbol{\xi}_{\boldsymbol{p}} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$.

The restriction of R to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is denoted by $R_{\boldsymbol{p}}$. The expression of $R_{\boldsymbol{p}}$ is specific to the considered manifold.

A special case of retraction is given by the *Exponential map*. An affine connection on $\mathcal{M}$ allows one to define the notion of a *geodesic* through the point $\boldsymbol{p}$ (see [KN96, §3.6]). We give here a intuitive definition of a geodesic. A geodesic on $\mathcal{M}$ is a curve $\gamma : \mathbb{R} \mapsto \mathcal{M}$ which has zero acceleration as seen from the manifold–we clarify this notion hereafter. In $\mathbb{R}^n$, geodesics are straight lines parametrized by arc length. On the sphere $\mathbb{S}^{n-1}$, geodesics are great circles, i.e., circles centred at the origin and with unit radius, also parametrized by arc length. The acceleration of γ at time t as seen from the manifold is the component of the acceleration vector $\gamma''(t)$ that is tangent to the manifold at $\gamma(t)$. This is very much linked to the fact that the acceleration of an object flying

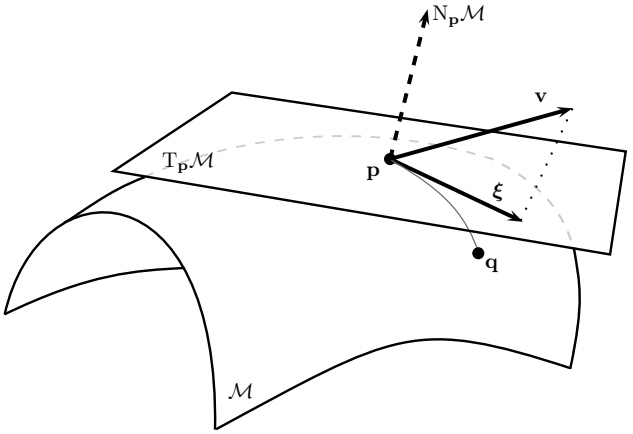


Figure 3.2: Illustration of some basic differential geometry tools on a smooth manifold embedded in $\mathbb{R}^n$. The tangent space at the point $\boldsymbol{p}$ is represented. The vector $\boldsymbol{v}$ is projected on $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ resulting in the tangent vector $\boldsymbol{\xi}$. The point $\boldsymbol{q}$ is obtained by taking the retraction of $\boldsymbol{\xi}$ at $\boldsymbol{p}$. The normal space at $\boldsymbol{p}$ is the orthogonal complement to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$.

along a great circle with constant speed is directed toward the center of the sphere, i.e., is always orthogonal to the tangent space. Hence, as seen from the sphere, the object has zero acceleration.

Let $\boldsymbol{\xi} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ be a tangent vector to the manifold at $\boldsymbol{p}$. There is a unique geodesic $\gamma(t)$ satisfying $\gamma(0) = \boldsymbol{p}$ with initial tangent vector $\gamma'(0) = \boldsymbol{\xi}$. The corresponding *Exponential map* is defined by

$$\mathrm{Exp}_{\boldsymbol{p}}(\boldsymbol{\xi}) = \gamma(1).$$

In general, the exponential map is only locally defined, that is, it only takes a neighbourhood of the origin in $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, to a neighbourhood of $\boldsymbol{p}$ in the manifold. This is because it relies on the theorem of existence and uniqueness for ordinary differential equations which is local in nature. An affine connection is called complete if the exponential map is well-defined at every point of the tangent bundle.

The exponential map is a common choice of retraction. However, computing the exponential map might be computationally inefficient, depending on the manifold setting. In this context, retractions can be seen as cheap alternatives to approximate the exponential map, while preserving its convergence properties in optimization methods.

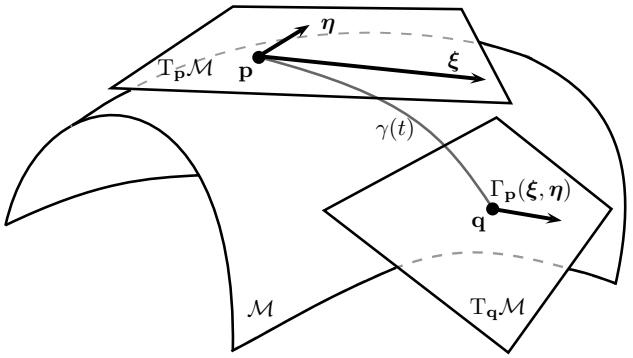


Figure 3.3: Illustration of the notions of geodesic, exponential map, logarithmic map and parallel transport. The geodesic $\gamma(t)$ is going through the points $\gamma(0) = \boldsymbol{p}$ and $\gamma(1) = \boldsymbol{q}$. Its initial velocity $\gamma'(0)$ is the vector $\boldsymbol{\xi}$ in the tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$. The point $\boldsymbol{q}$ is obtained by taking the exponential map $\mathrm{Exp}_{\boldsymbol{p}}(\boldsymbol{\xi})$. Conversely, taking the logarithmic map at $\boldsymbol{p}$ of $\boldsymbol{q}$ gives the tangent vector $\boldsymbol{\xi}$. The vector $\boldsymbol{\eta}$ in the tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is parallel transported to $\mathrm{T}_{\boldsymbol{q}}\mathcal{M}$, following the geodesic with initial velocity $\boldsymbol{\eta}$.

Given two points $\boldsymbol{p}$ and $\boldsymbol{q}$ on $\mathcal{M}$, one can compute the tangent vector $\boldsymbol{\xi} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ corresponding to the initial velocity of the geodesic $\gamma(t)$, going through $\boldsymbol{p}$ and $\boldsymbol{q}$ (i.e. $\gamma(0) = \boldsymbol{p}$, $\gamma(1) = \boldsymbol{q}$), using the corresponding *Logarithmic map*, defined by

$$\mathrm{Log}_{\boldsymbol{p}}(\boldsymbol{q}) = \boldsymbol{\xi} = \gamma'(0).$$

Finally, *parallel transport* is the tool allowing to transport a vector along smooth curves in a manifold. As illustrated in Figure 3.3, if the manifold is equipped with an affine connection, then it is possible to transport vectors along these curves so that they stay parallel with respect to the connection. Just as a retraction is an approximation of the exponential map, *vector transport* is a cheap alternative to parallel transport. More formally, let $T\mathcal{M} \oplus \mathrm{T}\mathcal{M}$ denote the set $\{(\eta_x, \xi_x) : \eta_x, \xi_x \in \mathrm{T}_x\mathcal{M}, x \in \mathcal{M}\}$. This set admits a natural manifold structure. A *vector transport* on a manifold $\mathcal{M}$ on top of a retraction R is a smooth map:

$$\mathrm{T}\mathcal{M} \oplus \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M} : (\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}(\xi_x) \in \mathrm{T}\mathcal{M}$$

satisfying the following properties for all $x \in \mathcal{M}$:

- (Underlying retraction) $\mathcal{T}_{\eta_x}(\xi_x)$ belongs to $\mathrm{T}_{R_x(\eta_x)}\mathcal{M}$;
- (Consistency) $\mathcal{T}_{0_x}(\xi_x) = \xi_x$ for all $\xi_x \in \mathrm{T}_x\mathcal{M}$;
- (Linearity) $\mathcal{T}_{\eta_x}(a\xi_x + b\zeta_x) = a\mathcal{T}_{\eta_x}(\xi_x) + b\mathcal{T}_{\eta_x}(\zeta_x)$.

3.3.2 Quotient matrix manifolds

As explained above, in a quotient matrix manifold, each point represents an equivalence class of matrices. Let $\mathcal{M}$ be a manifold equipped with an equivalence relation denoted $\sim$. The equivalence class of a point $\boldsymbol{p}$ on $\mathcal{M}$ (also called a *fiber*) is denoted by $[\boldsymbol{p}]$ and the notation $\boldsymbol{p} \sim \boldsymbol{q}$ means that these two points belong to the same equivalence class. A quotient manifold $\widetilde{\mathcal{M}}$ is then defined as the quotient of $\mathcal{M}$ by $\sim$:

$$\widetilde{\mathcal{M}} = \mathcal{M}/\sim = \{[\boldsymbol{p}] \ : \ \boldsymbol{p} \in \mathcal{M}\},$$

and $\mathcal{M}$ is called the *total space* of $\widetilde{\mathcal{M}}$. The mapping π from $\mathcal{M}$ to $\widetilde{\mathcal{M}}$ is known as the canonical projection. These concepts are illustrated in Figure 3.4.

Given a tangent vector $\boldsymbol{\xi}_{\boldsymbol{p}}$ at a point $\boldsymbol{p}$ in the total space $\mathcal{M}$, it is always possible to decompose it into two components $\boldsymbol{\xi}_{\boldsymbol{p}}^{\mathcal{V}}$ (the *vertical component*) and $\boldsymbol{\xi}_{\boldsymbol{p}}^{\mathcal{H}}$ (the *horizontal component*). The vertical component $\boldsymbol{\xi}_{\boldsymbol{p}}^{\mathcal{V}}$ belongs to the tangent space to the fiber going through $\boldsymbol{p}$. This decomposition into vertical and horizontal components aims at keeping all the directions that induce a displacement inside the equivalence class of $\boldsymbol{p}$ inside $\boldsymbol{\xi}_{\boldsymbol{p}}^{\mathcal{V}}$. The component $\boldsymbol{\xi}_{\boldsymbol{p}}^{\mathcal{H}}$ therefore only contains directions heading outside the equivalence class of $\boldsymbol{p}$. The tangent space of $\mathcal{M}$ at $\boldsymbol{p}$ is thus decomposed into the *vertical space* and the *horizontal space*, as follows:

$$\mathrm{T}_{\boldsymbol{p}}\mathcal{M} = \mathcal{V}_{\boldsymbol{p}}\mathcal{M} \oplus \mathcal{H}_{\boldsymbol{p}}\mathcal{M}.$$

However, just as a point $[\boldsymbol{p}] \in \widetilde{\mathcal{M}}$ is an abstract object and needs a representative in computational methods, a vector $\boldsymbol{\xi}_{[\boldsymbol{p}]} \in \mathrm{T}_{[\boldsymbol{p}]}\widetilde{\mathcal{M}}$ is also an abstract object. One way to uniquely represent a tangent vector $\boldsymbol{\xi}_{[\boldsymbol{p}]}$ is to choose a horizontal vector $\bar{\boldsymbol{\xi}}_{\boldsymbol{p}} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ such that its contribution through the projection π is the tangent vector $\boldsymbol{\xi}_{[\boldsymbol{p}]}$ itself. This vector is called the *horizontal lift* and must respect the following condition:

$$D\pi(\boldsymbol{p})[\bar{\boldsymbol{\xi}}_{\boldsymbol{p}}] = \boldsymbol{\xi}_{[\boldsymbol{p}]},$$

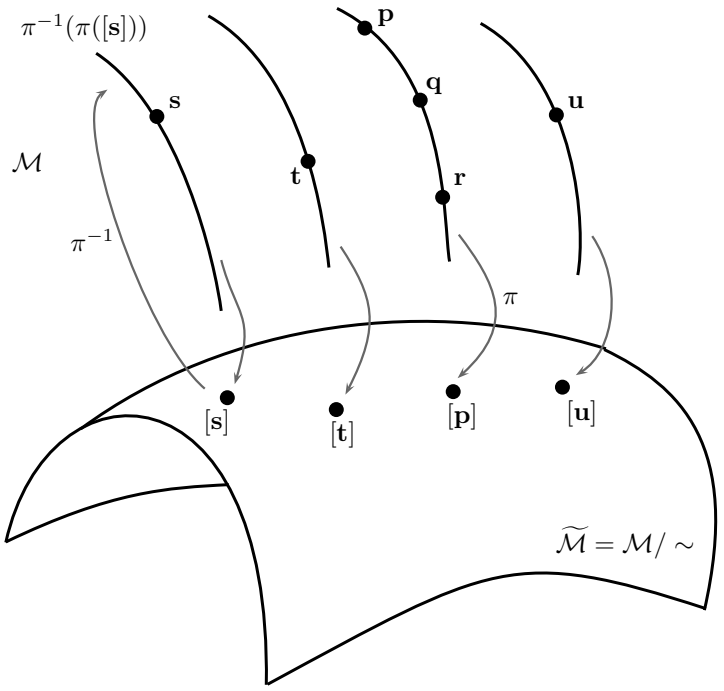


Figure 3.4: Illustration of the notions of quotient manifold, equivalence classes, fibers and total space. The points $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{t}$ and $\boldsymbol{u}$ lie on the manifold $\mathcal{M}$, the total space of $\widetilde{\mathcal{M}}$, equipped with an equivalence relation $\sim$. The points $\boldsymbol{p}, \boldsymbol{q}$ and $\boldsymbol{r}$ belong to the same equivalence class (fiber), i.e. $\boldsymbol{p} \sim \boldsymbol{q} \sim \boldsymbol{r}$. Note that points on a fiber are not necessarily spatially contiguous in $\mathcal{M}$, but conceptually contiguous through the relation $\sim$. The canonical projection π maps the points to their equivalence classes on $\widetilde{\mathcal{M}}$.

where $Df(\cdot)[\boldsymbol{d}]$ denotes the directional derivative of f in the direction $\boldsymbol{d}$.

The mapping $D\pi(\boldsymbol{p})$ restricted to $\mathcal{H}_p\mathcal{M}$, with its inverse given by the horizontal lift at $\boldsymbol{p}$, thus induces the identification

$$\mathrm{T}_{[\boldsymbol{p}]}\widetilde{\mathcal{M}} \simeq \mathcal{H}_{\boldsymbol{p}}\mathcal{M},$$

and makes it possible to represent abstract elements of $\mathrm{T}_{[\boldsymbol{p}]}\widetilde{\mathcal{M}}$ as elements of $\mathcal{H}_{\boldsymbol{p}}\mathcal{M}$. These concepts are illustrated in Figure 3.5.

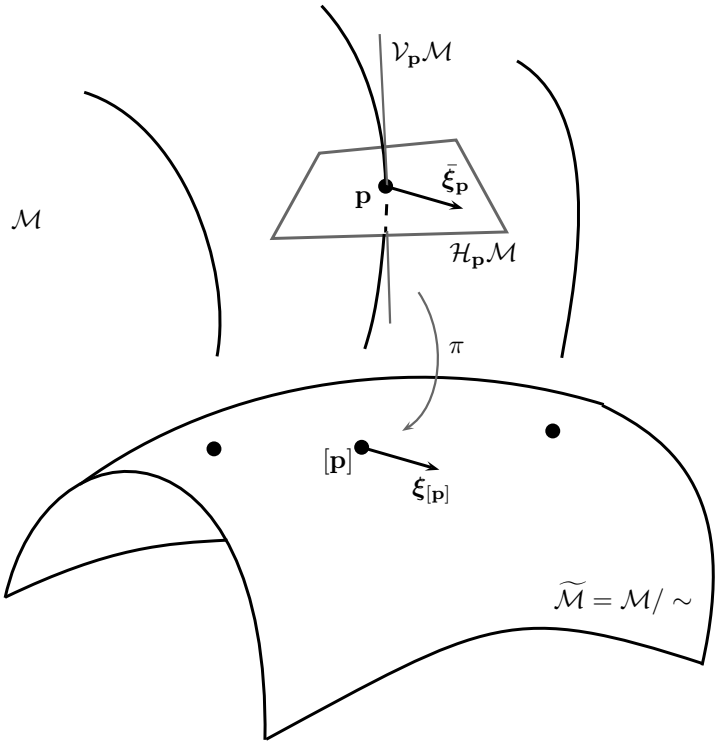


Figure 3.5: Illustration of the notion of tangent space for a quotient manifold. The horizontal and vertical spaces are displayed at the point $\boldsymbol{p} \in \mathcal{M}$. The horizontal lift $\bar{\boldsymbol{\xi}}_{\boldsymbol{p}}$ is used as the representative of the tangent vector $\boldsymbol{\xi}_{[\boldsymbol{p}]}$.

3.4 Optimization tools

3.4.1 Projected gradient

The notion of gradient can be defined on a Riemannian manifold. In the Euclidean case, the gradient of a smooth function f at a point $\boldsymbol{x}$ is linked to the directional derivative in the following way:

$$Df(\boldsymbol{x})[\boldsymbol{d}] = \nabla f(\boldsymbol{x})^\top \boldsymbol{d}.$$

Now, let $f : \mathcal{M} \mapsto \mathbb{R}$ be a function defined on a Riemannian submanifold of $\mathbb{R}^n$. Using the inner product $\langle ., . \rangle_{\boldsymbol{p}}$ defined in each tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, the gradient of f must respect a similar condition for any tangent vector $\boldsymbol{\xi}_{\boldsymbol{p}}$:

$$Df(\boldsymbol{x})[\boldsymbol{\xi}_{\boldsymbol{p}}] = \langle \operatorname{grad} f(\boldsymbol{x}), \boldsymbol{\xi}_{\boldsymbol{p}} \rangle .$$

The gradient $\operatorname{grad} f(\boldsymbol{x})$ must therefore belong to the tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$. If the inner product $\langle ., . \rangle_{\boldsymbol{p}}$ is defined as the restriction of the canonical inner product to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, then $\operatorname{grad} f(\boldsymbol{p})$ is simply computed as follows:

$$\operatorname{grad} f(\boldsymbol{x}) = P_{\boldsymbol{p}} \left(\nabla f(\boldsymbol{x}) \right),$$

as illustrated in Figure 3.6.

3.4.2 Steepest descent on matrix manifolds

Using the projected gradient described in the previous section, one can adapt the steepest descent algorithm presented in Section 2.4.1 to embedded manifolds settings.

Given an initial admissible point $\boldsymbol{p^0}$ on a manifold $\mathcal{M}$, the optimization scheme must be able to generate a sequence $\{\boldsymbol{p}^k\}_{k \geq 1}$ of iterates on the manifold. In particular, we would like to produce a new iterate $\boldsymbol{p}^{k+1}$ from the previous iterate $\boldsymbol{p}^k$ using a given direction $\boldsymbol{d}^k \in \mathbb{R}^n$ and a given step size $\alpha^k \in \mathbb{R}$. In the classical Euclidean case, this relationship is simply given by $\boldsymbol{p}^{k+1} = \boldsymbol{p}^k + \alpha^k \boldsymbol{d}^k$. In a manifold setting, this is obviously not a consistent rule in general, since the generated point $\boldsymbol{p}^{k+1}$ would probably not be on the manifold.

Using the tools presented in Section 3.3.1, one can adapt the directional iterative process $\boldsymbol{p}^{k+1} = \boldsymbol{p}^k + \alpha^k \boldsymbol{d}^k$ described above. Given an initial point $\boldsymbol{p}^0 \in \mathcal{M}$, a sequence $\{\boldsymbol{p}^k\} \subset \mathcal{M}$ can be constructed using the following expression:

$$\boldsymbol{p}^{k+1} = R_{\boldsymbol{p^k}} \left(\alpha^k P^k_{\boldsymbol{p}}(\boldsymbol{d}^k) \right), \ \alpha^k \in \mathbb{R}, \ \boldsymbol{d}^k \in \mathbb{R}^n.$$

In the context of a steepest descent step, where the direction $\boldsymbol{d}^k$ is given by the projected gradient $\operatorname{grad} f(\boldsymbol{p}^k)$, the line search procedure (e.g. using Armijo's rule) is then adapted, using the above update equation (as illustrated in Figure 3.6).

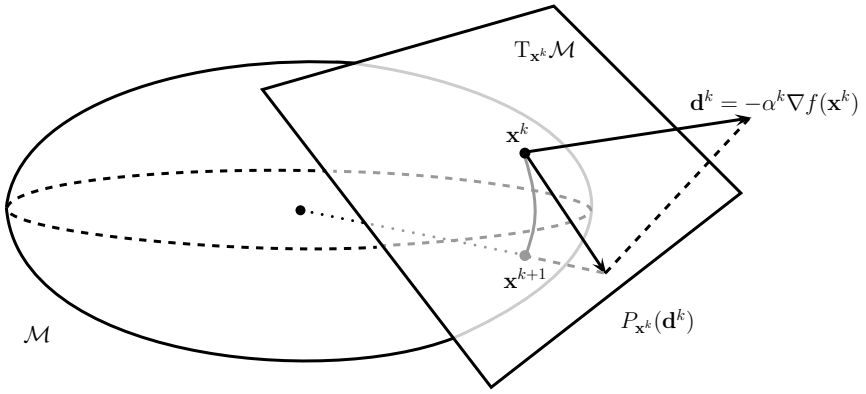


Figure 3.6: Illustration of a steepest descent step in a Riemannian manifold setting.

Part II

Applications

Chapter 4

Minimal Volume Oriented Bounding Box

Chapter overview

In this first applicative chapter, we study the Oriented Bounding Box (OBB) problem. This problem consists in finding the minimal-volume oriented bounding box (OBB) enclosing a given set of points in $\mathbb{R}^3$. Yet simple to state, this problem is computationally challenging, mainly due to the multimodal and nonsmooth characteristics of the associated cost function. Existing state-of-the-art methods dealing with this problem are either exact but slow, or fast but very approximative and unreliable.

In this work, we propose two methods based on the Particle Swarm Optimization (PSO) and the Differential Evolution (DE) metaheuristics to provide exploration of the multimodal search space. The original PSO and DE algorithms are modified so as to search for optimal solutions over the rotation group matrix manifold SO(3). The individuals in the populations of these algorithms are defined as 3D rotation matrices and operations are expressed over SO(3) using matrix products, exponentials and logarithms. The symmetry of the problem is also exploited.

Numerical experiments show that the proposed algorithms provide competitive trade-off between speed and accuracy, often outperforming existing methods.

Some of the material of this chapter appeared in the following publication:
[BA10] P.B. Borckmans and P.-A. Absil. Oriented Bounding Box Computation Using Particle Swarm Optimization. *In Proceedings of the 18th European Symposium on Artificial Neural Networks*, ESANN 2010.

4.1 Introduction

The Oriented Bounding Box problem (OBB) can be stated as follows: given a set of n points in $\mathbb{R}^3$, find the minimal-volume oriented parallelepiped enclosing all the points. This question arises in many practical applications and notably in computer vision problems. In collision detection for example, intersections are preferably checked using bounding volumes, such as boxes, spheres or ellipsoids, since it is computationally more efficient than doing so with complex 3D shapes (convex hull, ...). Oriented bounding boxes are a common choice because of the simplicity of the intersection test.

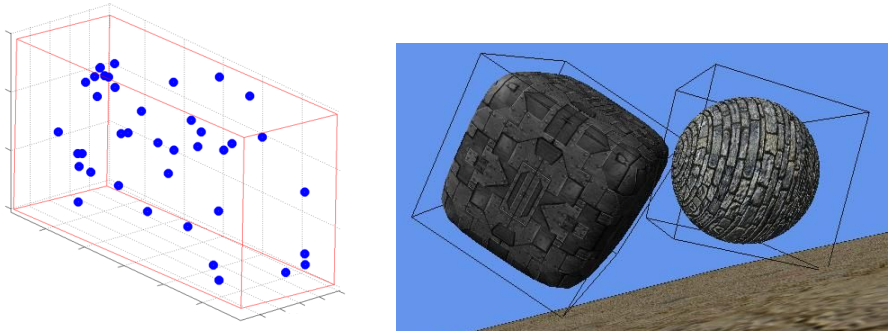


Figure 4.1: Illustration of the oriented bounding box problem and its application to objects intersection test.

However, computing an optimal OBB for a given set of points is not trivial. The best existing methods for solving the OBB problem can be sorted into two categories: exact methods and approximations. For 2-dimensional problems, one can compute an optimal bounding rectangle in linear time, using the so-called rotating callipers method, proposed by Toussaint in 1983 [Tou83]. An extension of this work for 3D instances has been proposed by O'Rourke in 1985 [O'R85] and is currently the best

exact algorithm. While this 3-dimensional adaptation is exact (thanks to an exhaustive research of possible solutions), it is hard to implement and it runs with a complexity of $\mathcal{O}(n^3)$, which is quite inefficient for large data sets, especially in applications requiring to perform the bounding box detection on the fly.

In the second category (approximations), some heuristics have been proposed using mainly principal component analysis (PCA) techniques [BHP99] or brute-force approaches [Eri04]. PCA based methods can be implemented easily and produce an approximation very quickly. However, their precision is very sensitive to the data point distribution and can result in far from optimal volumes [DKKR09]. Brute-force methods are usually based on a sampling of the search space; they are often inefficient, notably because they do not exploit the structure of the problem. For a more detailed state-of-the-art review of existing methods, the reader can refer to [CGM11].

4.2 Formulation of the problem

In this section, we provide the classical formulation of the OBB problem as an optimization problem and we present its reformulation over the rotation group SO(3) as proposed in [BA10] and [CGM11].

4.2.1 Classical formulation

Let $\mathcal{X}$ be the set of points $\boldsymbol{x}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i = 1, \ldots, n$, defining the geometry of the object one would like to enclose in a minimal volume oriented bounding box. The direct formulation as an optimization problem is the following:

$$\begin{aligned} \min_{a_1,a_2,a_3,\boldsymbol{x}^c,\boldsymbol{R}} \quad & a_1 \cdot a_2 \cdot a_3 \\ s.t. \quad & -\tfrac{\boldsymbol{a}}{2} \leq \boldsymbol{R}(\boldsymbol{x}_i - \boldsymbol{x}^c) \leq \tfrac{\boldsymbol{a}}{2}, \qquad \forall i \in \{1, \cdots, N\} \\ & \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I} \end{aligned}$$

where the vector $\boldsymbol{a} = (a_1, a_2, a_3) \in \mathbb{R}^3$ denotes the lengths of the sides of the box, $\boldsymbol{x}^c \in \mathbb{R}^3$ is its center, $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix giving its orientation and $N = |\mathcal{X}|$, i.e. the number of points in the set of points $\mathcal{X}$ (or equivalently the points defining its convex hull, since only those points are relevant for the computation of the bounding box). The inequality in the first constraint is to be taken component-wise.

The objective function is trilinear (linear in each of the variables a_1, a_2 and a_3). Furthermore, the three first constraints, ensuring that all the points lie inside the box, are also linear. The main difficulty lies in the last constraint: $\boldsymbol{R}$ is a rotation matrix, thus it is orthogonal. The minimization over the 6 scalar variables and the rotation matrix can in fact be split in two successive minimizations, as proposed in [BA10] and [CGM11]. Using the rotation group:

$$\mathrm{SO}(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} : \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \det(\boldsymbol{R}) = 1\},$$

the minimization for $\boldsymbol{R}$ is given by

$$\min_{\boldsymbol{R}\in \mathrm{SO}(3)} f(\boldsymbol{R}), \tag{4.1}$$

where $f(\boldsymbol{R})$ is given by the solution of the subproblem over a_1, a_2, a_3 and $\boldsymbol{x}^c$ for a given matrix $\boldsymbol{R}$. Let $x'_{min}, x'_{max}, y'_{min}, y'_{max}, z'_{min}, z'_{max}$ be the minimal and maximal coordinates in each dimension of the set of points in $\mathcal{X}$ after applying the rotation $\boldsymbol{R}$. The function $f(\boldsymbol{R})$ can be expressed as:

$$f(\boldsymbol{R}) = (x'_{\max} - x'_{\min})(y'_{\max} - y'_{\min})(z'_{\max} - z'_{\min}), \tag{4.2}$$

and corresponds to the volume of the axis-aligned bounding box, in the reference frame rotated by $\boldsymbol{R}$.

4.2.2 Formulation on SO(3)

The idea presented in this chapter is to exploit this reformulation of the OBB problem as an optimization problem over SO(3). As can be observed with a 2D example in Figure 4.2, the function $f(R)$ is only $\mathcal{C}^0$ and presents multiple minima. The non-differentiable and multimodal aspects of $f(\boldsymbol{R})$ make it a good candidate for derivative-free optimization methods and global exploration methods [CSV09b]. In [CGM11], the authors propose a hybridization of the genetic and the Nelder-Mead algorithms to solve this optimization problem. Mixing these two algorithms ensures a good balance between exploration of the search space and refinement of potential optima.

In this chapter, we propose to adapt two metaheuristics to this new formulation of the OBB problem: the Particle Swarm Optimization

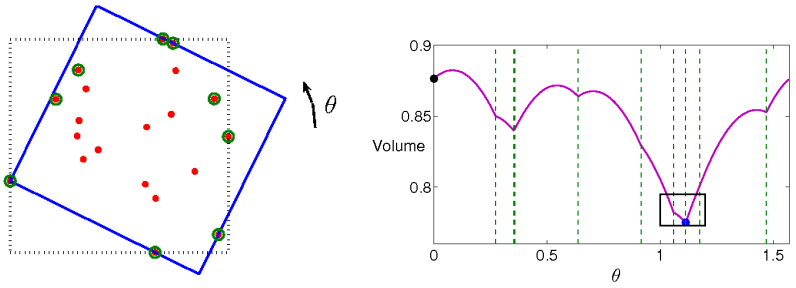


Figure 4.2: 2D Oriented bounding box: the volume $f(\boldsymbol{R})$ is not differentiable everywhere and presents multiple minima.

(PSO) and the Differential Evolution (DE) algorithms. While the authors of [CGM11] focused on obtaining high quality numerical results, the intent of the present work is to provide generic ways of adapting these two methods to manifold settings. These two methods are good solver candidates since they are clearly suited for non-differentiable, multimodal problems. PSO is known to exhibit intrinsic qualities to find a good trade-off between exploration and exploitation of the search domain. DE on the other hand is often praised for its global exploration capabilities. In order to achieve the adaptation of these algorithms for the OBB problem, they need to be modified to fit the SO(3) search space. Some properties of this manifold are used to keep the proposed methods as close as possible to the original algorithms. The main contribution of this chapter is to provide proof-of-concept adaptations of these two metaheuristics to the special orthogonal group. Nevertheless, the framework presented below can easily be generalized for any Riemannian manifold and the key ingredients of the metaheuristics considered here are common to many other techniques.

The next sections are organized as follows: Section 4.3 recalls the PSO algorithm basics and presents its adaptation to the special orthogonal group SO(3). Section 4.4 discusses some symmetry properties of the OBB problem and their impact on the algorithmic design. Section 4.5 recalls the DE algorithm and details how it can be used in a manifold setting. Section 4.6 presents some numerical results, showing that the proposed methods are faster and/or more accurate than existing meth-

ods. Section 4.7 exposes possible enhancements of the methods and other ideas that could be further investigated.

4.3 PSO for the OBB problem

4.3.1 Particle Swarm Optimization

We recall here the principles of the PSO algorithm presented in Chapter 2, Section 2.4.3. First introduced in 1995 by Eberhart and Kennedy [KE95], PSO is a stochastic population-based algorithm. Initially presented as a tool to simulate the coordination of individuals in fishes and birds populations, PSO was rapidly adopted by the optimization heuristics community. Particles are points evolving in the search space, following simple rules, mimicking the behaviour of social groups. The emerging behaviour of the entire swarm can be described as an oscillating movement converging to a single point. Although there is no theoretical guarantee that this point is a local minimizer of the objective function, experimentation shows that is often the case in practice. Furthermore, simple enhancements of the original PSO method (such as the GCPSO variant presented in Chapter 5) allow to provide such guarantees.

The population (called the swarm) is a set of points. These points are initialized randomly in $\mathbb{R}^n$ and the driving force of the optimization process is given by the following update equations (iterated over k), for each particle (indexed by i):

$$\boldsymbol{v}_i(k+1) = \underbrace{w(k)\boldsymbol{v}_i(k)}_{\text{inertia}} + \underbrace{c\alpha_i(k)\left(\boldsymbol{y}_i(k) - \boldsymbol{x}_i(k)\right)}_{\text{nostalgia}} + \underbrace{s\beta_i(k)\left(\hat{\boldsymbol{y}}(k) - \boldsymbol{x}_i(k)\right)}_{\text{social}} \tag{4.3}$$

$$\boldsymbol{x}_i(k+1) = \boldsymbol{x}_i(k) + \boldsymbol{v}_i(k+1)\ , \tag{4.4}$$

where $\boldsymbol{x}_i$ denotes the position of the i-th particle, $\boldsymbol{v}_i$ denotes its velocity, $\boldsymbol{y}_i$ is its personal best position so far and $\hat{\boldsymbol{y}}$ is the global best position of the swarm so far. The inertia coefficient w adjusts the global exploration tendency of the swarm and is usually dynamic (for instance linearly decreasing from 1 to 0, from iteration 1 to last iteration). The coefficients c and s are two adjustable parameters (called the cognition and social factors). Typical values for c and s are in the range $[0.5, 2]$. Finally, α and β are random components (usually distributed as $\sim U(0,1)$), also called "craziness" in the PSO literature. As can be seen, the behaviour of each particle is dictated by velocity increments composed of three simple components: inertia, cognition (nostalgic behaviour) and social

behaviour (tendency to follow the leader). At the end of each iteration, $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}$ are updated.

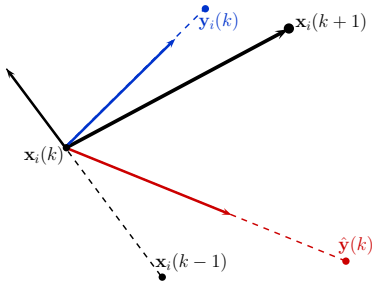


Figure 4.3: Illustration of the update rules for the PSO algorithm. The thick black arrow represents the position update and is obtained as the weighted sum of the thin black vector (inertia term), the blue vector (nostalgia term) and the red (social term) vector, according to (4.4)

The PSO method has gained a lot of attention since the late 1990's thanks to its interesting exploration properties. Many variations of this initial implementation are available nowadays, providing specific features such as better local convergence or niching capabilities. In this work, the main intent is to detail the adaptation of metaheuristics to SO(3) and we therefore implement the classical PSO algorithm for the sake of clarity. For more in-depth information about PSO and its many variations, the reader is referred to [Eng06].

4.3.2 Adapting PSO to SO(3)

In its original form, PSO is described for particles distributed in $\mathbb{R}^n$ so that $\boldsymbol{x}_i$, $\boldsymbol{y}_i$, $\hat{\boldsymbol{y}}$ and $\boldsymbol{v}_i \in \mathbb{R}^n$ and the operations involved in the update equations $(+,-,.)$ are the usual vectorial addition, difference and scaling. In order to adapt the standard PSO method to the SO(3) search space, one must redefine $\boldsymbol{x}_i$, $\boldsymbol{v}_i$ and the operations mentioned above. Since we are looking for the optimal rotation matrix, the position $\boldsymbol{X}_i$ must be an element of SO(3) and so do $\boldsymbol{Y}_i$ and $\hat{\boldsymbol{Y}}$. The velocity $\boldsymbol{V}_i$ is now an element of the tangent space $\mathrm{T}_{\boldsymbol{X}_i}\mathrm{SO}(3)$ to SO(3) centred at $\boldsymbol{X}_i$. Since SO(3) is a Lie group (see e.g. [Boo03]), we have $\boldsymbol{V}_i = \boldsymbol{X}_i\tilde{\boldsymbol{V}}_i$ for some $\tilde{\boldsymbol{V}}_i \in \mathfrak{so}(3)$, where $\mathfrak{so}(3) = \mathrm{T}_{\boldsymbol{I}}\mathrm{SO}(3)$ is the set of all skew-symmetric 3×3 matrices. Noting that the geodesic $\gamma(t)$ on SO(3) with initial position $\boldsymbol{X}_i \in \mathrm{SO}(3)$

and initial velocity $\boldsymbol{X}_i\tilde{\boldsymbol{V}}_i$ is given by $\gamma(t) = \mathrm{Exp}_{\boldsymbol{X}_i}(\boldsymbol{V}_i) = \boldsymbol{X}_i \exp(t\tilde{\boldsymbol{V}}_i)$ (see [Boo03, §VII.8]), the position update in the second equation 4.4 can be rewritten using matrix composition:

$$\begin{aligned} \boldsymbol{X}_i(k+1) &= \mathrm{Exp}_{\boldsymbol{X}_i(k)}\left(\boldsymbol{V}_i(k+1)\right) \\ &= \boldsymbol{X}_i(k)\exp(\tilde{\boldsymbol{V}}_i(k+1)). \end{aligned} \tag{4.5}$$

Moreover, given two points $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ in SO(3), the initial velocity $\boldsymbol{V} = \boldsymbol{X}_1\tilde{\boldsymbol{V}}$ of the geodesic $\gamma(t)$ starting from $\boldsymbol{X}_1$ that goes through $\boldsymbol{X}_2$ at $t = 1$ is given by $\boldsymbol{V} = \mathrm{Log}_{\boldsymbol{X}_1}(\boldsymbol{X}_2) = \boldsymbol{X}_1\tilde{\boldsymbol{V}} = \boldsymbol{X}_1\log(\boldsymbol{X}_1^\top\boldsymbol{X}_2)$. Exploiting this expression for the nostalgia and social components of the velocity equation 4.3, one can write:

$$\begin{aligned} \tilde{\boldsymbol{V}}_i(k+1) = w(k)\tilde{\boldsymbol{V}}_i(k) \;+\; c\alpha_i(k)\log\left(\boldsymbol{X}_i(k)^\top\boldsymbol{Y}_i(k)\right) \\ + s\beta_i(k)\log\left(\boldsymbol{X}_i(k)^\top\hat{\boldsymbol{Y}}(k)\right). \end{aligned} \tag{4.6}$$

As figure 4.4 shows, the difference with classical PSO is that this combination now occurs in the tangent space centred at $\boldsymbol{X}(k)$, and is then retracted to the SO(3) manifold.

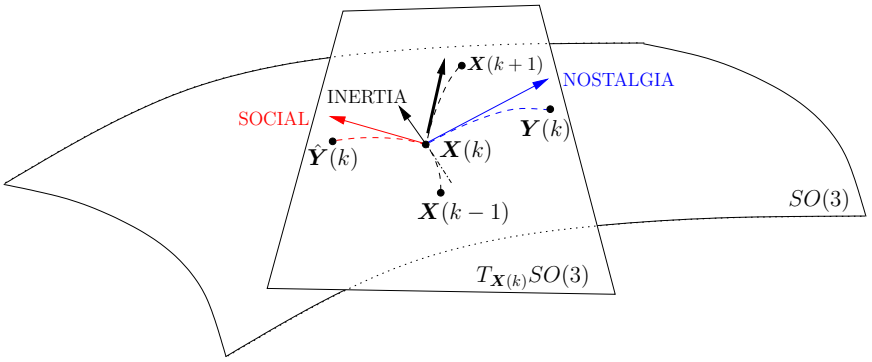


Figure 4.4: Computation of the PSO position update in the tangent space of SO(3) centred at $\boldsymbol{X}(k)$. The new iterate $\boldsymbol{X}(k+1)$ is constructed using a combination of inertia, nostalgia and social contributions in $\mathrm{T}_{\boldsymbol{X}(k)}\mathrm{SO}(3)$ and is then retracted to SO(3).

Note that the key idea behind the original PSO algorithm is preserved with this adaptation: the new rotation matrix is built using a

stochastically weighted combination of the old rotation (with inertia) and attraction towards both personal and global best rotations. Figure 4.5 illustrates the behaviour of the swarm on the SO(3) group.

4.4 Symmetry of the problem

Now that the update equations have been modified, the last step needed in order for PSO to be adapted to the SO(3) manifold is to take the symmetry of the OBB problem into account. Considering the 2-dimensional problem first, a rotation matrix belongs to SO(2) and can therefore be represented with a unique angle θ. Given such an orientation of a bounding rectangle, the corresponding area will clearly be the same for every further rotation of 90 degrees. This indicates that the search space should be truncated to the first quadrant only (matrices with orientation in $[0, \frac{\pi}{2}[$), using the relation: $\theta' = \theta \mod (\frac{\pi}{2})$. Furthermore, when computing the angle between two rotation matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, represented by angles θ_1 and θ_2, the shortest distance in the light of this periodicity should be exploited, using the following computation, so that the measured angle always lies in $]\frac{-\pi}{4}, \frac{\pi}{4}]$:

$$\Theta(\boldsymbol{R}_1, \boldsymbol{R}_2) = \begin{cases} \theta_2 - \theta_1, & \text{if } |\theta_2 - \theta_1| \leq \frac{\pi}{4} \\ \theta_2 - \theta_1 - \frac{\pi}{2}\text{sgn}(\theta_2 - \theta_1), & \text{if } |\theta_2 - \theta_1| > \frac{\pi}{4} \end{cases}$$

These properties can be regarded another way. The rows $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ of a matrix $\boldsymbol{R}$ in SO(2) form an orthogonal basis in $\mathbb{R}^2$. Considering all the 2D signed permutation matrices $\boldsymbol{P}$ in SO(2) (composed of columns $\boldsymbol{p}_i \in \{\pm\boldsymbol{e}_1, \pm\boldsymbol{e}_2\}$), $\boldsymbol{R}' = \boldsymbol{R}\boldsymbol{P}$ yields 4 different bases, each time with $\boldsymbol{r}'_1$ and $\boldsymbol{r}'_2$ lying in two consecutive quadrants. Since the log mapping $\log(\boldsymbol{R})$ measures the "displacement" between the identity matrix $\boldsymbol{I}$ and $\boldsymbol{R}$, choosing the permutation $\boldsymbol{P}$ that brings $\boldsymbol{r}'_1$ as close as possible to $\boldsymbol{e}_1$ and $\boldsymbol{r}'_2$ to $\boldsymbol{e}_2$ ensures that the smallest displacement $\boldsymbol{D}$ linking two matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ can be computed using $\boldsymbol{D} = \log(\boldsymbol{R}_1^\top \boldsymbol{R}_2 \boldsymbol{P})$. Figure 4.6 represents these symmetry properties.

This can now be extended to the 3-dimensional case. The rows $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ of $\boldsymbol{R}$ now form a basis of $\mathbb{R}^3$, and with the set of all 3D signed permutations $\boldsymbol{P}$ in SO(3), $\boldsymbol{R}' = \boldsymbol{R}\boldsymbol{P}$ yields 24 bases. The permutation can now be chosen so that $\boldsymbol{r}'_1$ is as close as possible to $\boldsymbol{e}_1$, $\boldsymbol{r}'_2$ to $\boldsymbol{e}_2$ and $\boldsymbol{r}'_3$ to $\boldsymbol{e}_3$, and the relation $\boldsymbol{D} = \log(\boldsymbol{R}_1^\top \boldsymbol{R}_2 \boldsymbol{P})$ still gives the smallest displacement linking $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$.

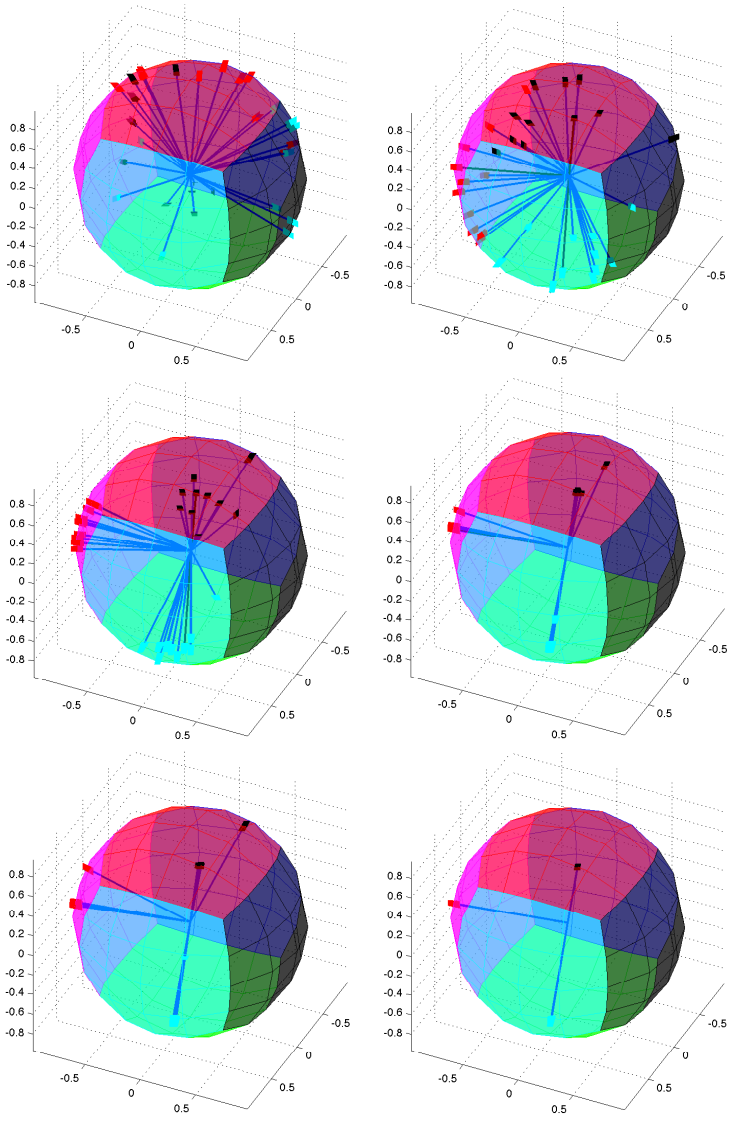


Figure 4.5: Illustration of the PSO algorithm on SO(3). Each triplet of axes (with red, black and cyan endpoints) represents a particle (a rotation matrix). Initially spread randomly across the set of rotation matrices, the swarm explores the search space before converging to a unique rotation matrix.

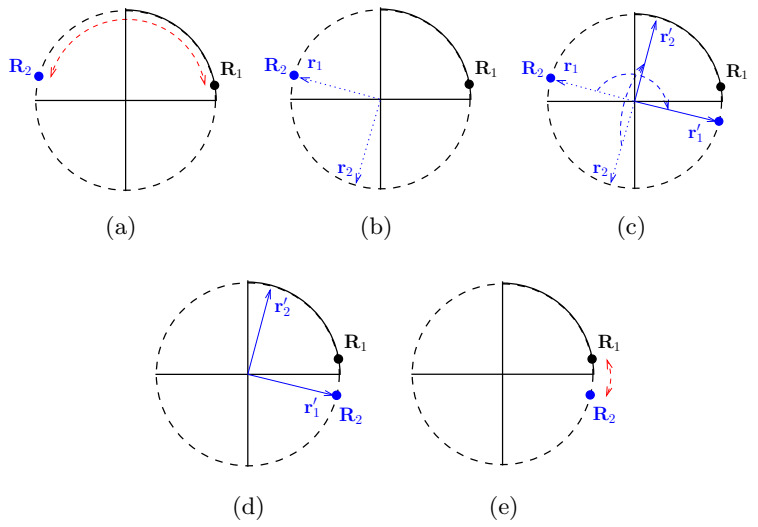


Figure 4.6: Illustration of the symmetry properties of the OBB problem in SO(2). (a) Measuring the angle between $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ without considering the symmetry properties. (b) Representation of the columns $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ of the matrix $\boldsymbol{R}_2$. (c) Choice of a signed permutation of the columns of $\boldsymbol{R}_2$ bringing them closer to those of $\boldsymbol{R}_1$. (d) An equivalent representation of $\boldsymbol{R}_2$. (e) Measuring the smallest displacement between $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$.

Taking the periodicity and symmetry into account, the update equations (4.5) and (4.6) can finally be written as:

$$\begin{cases} \tilde{\boldsymbol{V}}_i(k+1) &= w(k)\tilde{\boldsymbol{V}}_i(k) + c\alpha_i(k)\log\left(\boldsymbol{X}_i(k)^\top \boldsymbol{Y}_i(k)\boldsymbol{P}_1\right) \\ & \qquad\qquad + s\beta_i(k)\log\left(\boldsymbol{X}_i(k)^\top \hat{\boldsymbol{Y}}(k)\boldsymbol{P}_2\right), \\ \boldsymbol{X}_i(k+1) &= \boldsymbol{X}_i(k)\exp(\tilde{\boldsymbol{V}}_i(k+1)), \end{cases}$$

where $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are permutation matrices chosen as described above.

While this section described how symmetries of the OBB problem can be handled theoretically, we did not observe benefit while implementing this strategy. Therefore, the numerical results presented at the end of this chapter were computed without handling the symmetry. The absence of influence of these aspects remain to be investigated in further works.

4.5 DE for the OBB problem

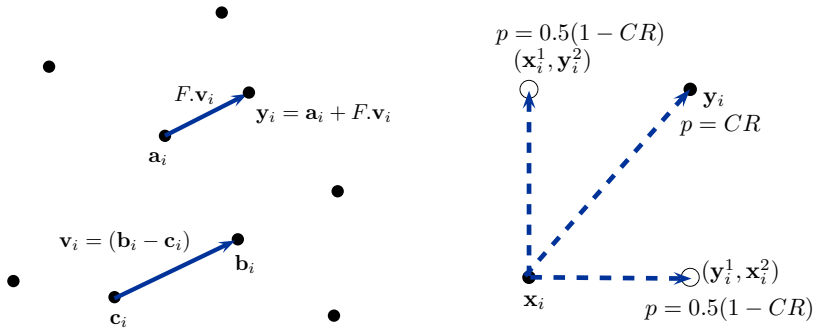
4.5.1 Differential Evolution

In this section, we recall the principles of the Differential Evolution algorithm presented in Section 2.4.3. The Differential Evolution algorithm (DE) maintains a population of candidate solutions in the search space and creates new candidate solutions by combining existing ones, according to simple rules. Originally due to Storn and Price [SP97] in 1997, this technique has gained popularity and several books have been published on theoretical and practical aspects of using DE in parallel computing, multiobjective optimization and constrained optimization (see e.g. [PSL05], [Cha08], [Feo06]).

The algorithm is initialized with a random population ($\boldsymbol{x}_i,\ i = 1,\ldots,m$) in the search space. Two parameters are defined:

- The differential weight $F \in [0, 2]$
- The crossover probability $\mathrm{CR} \in [0, 1]$.

As illustrated in Figure 4.7, each iteration (indexed by k) of the DE algorithm consists in two main steps for each individual $\boldsymbol{x}_i(k)$, namely the mutation and the crossover steps.



(a) Mutation (b) Crossover

Figure 4.7: Illustration of the DE algorithm update equations (mutation and crossover).

The purpose of the mutation step is to generate a new point $\boldsymbol{y}_i(k)$ that is a combination of three individuals randomly selected among the

population ($\boldsymbol{a}_i(k)$, $\boldsymbol{b}_i(k)$ and $\boldsymbol{c}_i(k)$). This combination is influenced by the differential weight parameter F:

$$\boldsymbol{y}_i(k) = \boldsymbol{a}_i(k) + F.\left(\boldsymbol{b}_i(k) - \boldsymbol{c}_i(k)\right) \tag{4.7}$$

Once this point is created, the crossover step is used to create a candidate point $\boldsymbol{z}_i(k)$. This point is set to be equal to the current iterate $\boldsymbol{x}_i(k)$ except for some components (denoted $\boldsymbol{z}(k)^j$) that are replaced by components of the point $\boldsymbol{y}_i(k)$ (denoted $\boldsymbol{y}(k)^j$) generated during the mutation step. The choice of which components are to be replaced is based on the crossover probability CR. Furthermore, one of the components (of index $j = R$ selected randomly) is always replaced to ensure that the candidate point $\boldsymbol{z}_i(k)$ is different from the current iterate $\boldsymbol{x}_i(k)$. Component-wise, this candidate point $\boldsymbol{z}_i(k)$ is thus computed as follows:

$$\boldsymbol{z}_i(k)^j = \begin{cases} \boldsymbol{y}_i(k)^j & \text{if CR} > r \sim U(0,1) \text{ or if } j = R, \\ \boldsymbol{x}_i(k)^j & \text{otherwise.} \end{cases} \tag{4.8}$$

Another way of expressing this operation (that will be useful for adapting the DE algorithm to the SO(3) manifold in the next section) is the following:

$$\boldsymbol{z}_i(k) = \boldsymbol{x}_i + F.\left(\sum_{j=1}^{n} \delta(k)_j.(\boldsymbol{y}_i(k)^j - \boldsymbol{x}_i(k)^j).\boldsymbol{e}_j\right), \tag{4.9}$$

where n is the dimension of the problem, $\boldsymbol{e}_j$ denotes the j-th canonical vector and $\boldsymbol{\delta}(k)$ controls which components are implicated in the crossover process:

$$\delta(k)_j = \begin{cases} 1 & \text{if CR} > r \sim U(0,1) \text{ or if } j = R, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the next iterate $\boldsymbol{x}_i(k+1)$ is set to $\boldsymbol{z}_i(k)$ if the corresponding objective value is improved or remains unchanged otherwise:

$$\boldsymbol{x}_i(k+1) = \begin{cases} \boldsymbol{z}_i(k) & \text{if } f(\boldsymbol{z}_i(k)) < f(\boldsymbol{x}_i(k)), \\ \boldsymbol{x}_i(k) & \text{otherwise.} \end{cases} \tag{4.10}$$

4.5.2 Adapting DE to SO(3)

Using similar mechanisms to those presented in Section 4.3.2, we now describe how the DE algorithm can be adapted to the rotation group SO(3). The individuals $\boldsymbol{X}_i$ and the points $\boldsymbol{Y}_i$, $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, $\boldsymbol{C}_i$ and $\boldsymbol{Z}_i$ used in the mutation and crossover steps are now elements of SO(3).

Mutation step

In order to generate the point $\boldsymbol{Y}_i$ in the mutation step described by (4.7), one first has to compute the tangent vector $\boldsymbol{V}_i \in \mathrm{T}_{\boldsymbol{C}_i}\mathrm{SO}(3)$ which corresponds to the initial velocity of the geodesic $\gamma(t)$ linking the points $\gamma(0) = \boldsymbol{C}_i$ and $\gamma(1) = \boldsymbol{B}_i$:

$$\boldsymbol{V}_i = \mathrm{Log}_{\boldsymbol{C}_i}(\boldsymbol{B}_i) = \boldsymbol{C}_i\tilde{\boldsymbol{V}}_i = \boldsymbol{C}_i \log\left(\boldsymbol{C}_i^\top \boldsymbol{B}_i\right).$$

Now, in order to use this tangent vector at the point $\boldsymbol{A}_i$ as implied by equation (4.7), it must be transported to $\mathrm{T}_{\boldsymbol{A}_i}\mathrm{SO}(3)$:

$$\boldsymbol{W}_i = \boldsymbol{A}_i\boldsymbol{C}_i^\top\boldsymbol{V}_i = \boldsymbol{A}_i\tilde{\boldsymbol{V}}_i.$$

Note that in a Lie group like SO(3), this step is trivial and might seem unnecessary. However, for other manifolds, one would need to use appropriate vector transport of $\boldsymbol{V}_i$ from $\mathrm{T}_{\boldsymbol{C}_i}\mathcal{M}$ to $\mathrm{T}_{\boldsymbol{A}_i}\mathcal{M}$ (as will be done in Chapter 5 for instance). Using matrix composition, the mutation step equation (4.7) can finally be rewritten as follows and is depicted in Figure 4.8:

$$\begin{aligned}\boldsymbol{Y}_i(k) &= \mathrm{Exp}_{\boldsymbol{A}_i}\left(F.\boldsymbol{W}_i\right) = \boldsymbol{A}_i \exp\left(F.\tilde{\boldsymbol{V}}_i\right)\\ &= \boldsymbol{A}_i \exp\left(F.\log\left(\boldsymbol{C}_i^\top\boldsymbol{B}_i\right)\right).\end{aligned} \tag{4.11}$$

Crossover step

The crossover step as defined in the original DE algorithm involves the mixing of some entries of two vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ to build the candidate point $\boldsymbol{z}_i$ (see (4.8)). While this construction is perfectly suitable in $\mathbb{R}^n$, it is clearly inappropriate in a manifold setting like SO(n), since exchanging entries from two rotation matrices would normally not lead to another valid rotation matrix. The reinterpretation (4.9) of the crossover

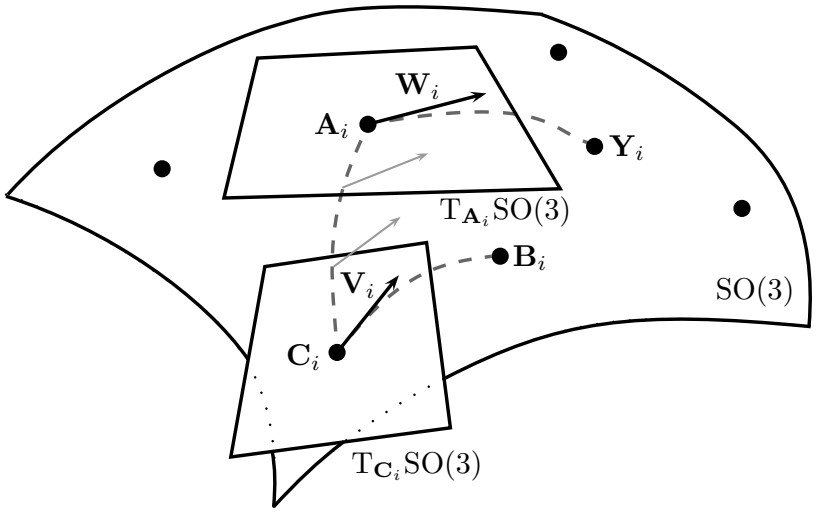


Figure 4.8: Adaptation of the DE mutation step on SO(3). The tangent vector $\boldsymbol{V}_i = \boldsymbol{C}_i \log\left(\boldsymbol{C}_i^\top \boldsymbol{B}_i\right)$ is transported to $\mathrm{T}_{\boldsymbol{A}_i}\mathrm{SO}(3)$. This transported vector $\boldsymbol{W}_i$ is then used to compute the candidate point $\boldsymbol{Y}_i$ using the exponential map.

equation (4.8) allows one to adapt the crossover step to a manifold setting.

In order to do so, a tangent vector $\boldsymbol{V}$ at a point $\boldsymbol{X}$ of the manifold must be decomposed using a basis of the vector space of the tangent space $\mathrm{T}_{\boldsymbol{X}}\mathcal{M}$:

$$\boldsymbol{V} = \sum_{j=1}^{n} \alpha_j \boldsymbol{E}_j,$$

where n is the dimension of the manifold and the set of elements $\{\boldsymbol{E}_j\}$ forms a basis of $\mathrm{T}_{\boldsymbol{X}}\mathcal{M}$, i.e. $\mathrm{T}_{\boldsymbol{X}}\mathcal{M} = \operatorname{span}\{\boldsymbol{E}_j\}$.

In the case of SO(3), a basis of $\mathrm{T}_{\boldsymbol{I}}\mathrm{SO}(3) = \mathfrak{so}(3)$ is simply given by:

$$\begin{aligned}\mathrm{T}_{\boldsymbol{I}}\mathrm{SO}(3) &= \operatorname{span}\{\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{S}_3\} \\ &= \operatorname{span}\left\{ \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \right\},\end{aligned} \tag{4.12}$$

and therefore, a tangent vector $\boldsymbol{V}$ at any point $\boldsymbol{X} \in \mathrm{SO}(3)$ can be

decomposed as follows (using the Lie algebra structure of SO(3)):

$$\boldsymbol{V} = \sum_{j=1}^{3} \alpha_j \boldsymbol{X} \boldsymbol{S}_j. \tag{4.13}$$

Noting that the initial velocity of the geodesic going through $\boldsymbol{X}_i(k)$ at $t = 0$ and $\boldsymbol{Y}_i(k)$ at $t = 1$ is given by

$$\boldsymbol{U}_i(k) = \mathrm{Log}_{\boldsymbol{X}_i(k)} \left(\boldsymbol{Y}_i(k) \right) = \boldsymbol{X}_i \log \left(\boldsymbol{X}_i(k)^\top \boldsymbol{Y}_i(k) \right),$$

one can thus obtain the following decomposition of this vector:

$$\boldsymbol{U}_i(k) = \sum_{j=1}^{3} \alpha_j \boldsymbol{X}_i(k) \boldsymbol{S}_j. \tag{4.14}$$

The crossover step can finally be adapted to SO(3) using the modified crossover equation (4.9) and the decomposition (4.14):

$$\begin{aligned} \boldsymbol{Z}_i(k) &= \mathrm{Exp}_{\boldsymbol{X}_i(k)} \left(\sum_{j=1}^{3} \alpha_j \delta(k)_j \boldsymbol{X}_i(k) \boldsymbol{S}_j \right) \\ &= \boldsymbol{X}_i(k) \exp \left(\sum_{j=1}^{3} \alpha_j \delta(k)_j \boldsymbol{S}_j \right), \end{aligned} \tag{4.15}$$

where $\boldsymbol{S}_j$ denotes the j-th element of the basis of $\mathrm{T}_{\boldsymbol{I}}\mathrm{SO}(3) = \mathfrak{so}(3)$ defined in (4.12) and $\boldsymbol{\delta}(k)$ controls which components are implicated in the crossover process:

$$\delta(k)_j = \begin{cases} 1 & \text{if CR} > r \sim U(0,1) \text{ or if } j = R, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 4.9 depicts the crossover step in a Riemannian manifold setting. For the sake of clarity, the illustration represents a 2-dimensional manifold but the concepts are easily extendible to higher dimensions.

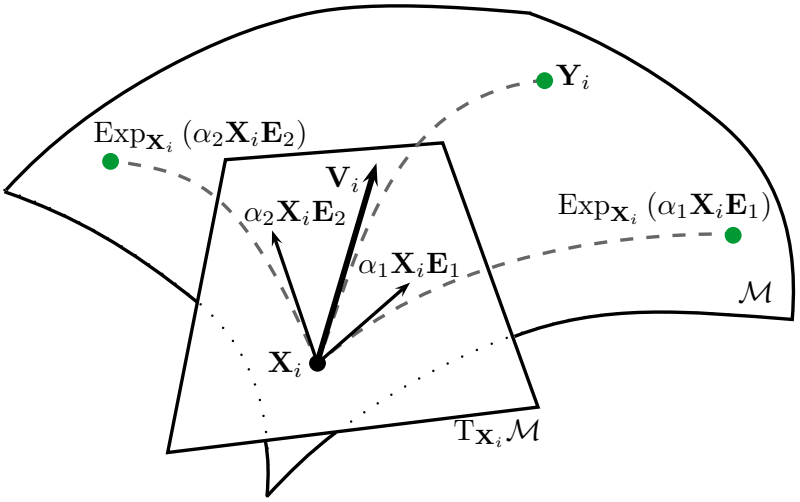


Figure 4.9: Adaptation of the DE crossover step on a 2-dimensional Riemannian manifold. The tangent vector $\boldsymbol{V}_i$ is decomposed in a basis of $\mathrm{T}_{\boldsymbol{X}_i}\mathcal{M} = \mathrm{span}\{\boldsymbol{X}_i\boldsymbol{E}_1, \boldsymbol{X}_i\boldsymbol{E}_2\}$. The three possible outcomes of the crossover step are depicted (green dots).

4.6 Numerical experiments

As preliminary experiments, the proposed adaptations of the PSO and DE algorithms were implemented using Matlab. Some data sets (available at `http://sites.uclouvain.be/absil/borckmans/OBB`) were chosen presenting various sizes (number of nodes on the convex hull) and distributions. Tables 4.1 and 4.2 presents the results of these simulations for eight of these data sets and compares the performance of the proposed solutions to two state-of-the-art methods: All-PCA and O'Rourke. The optimal volume is the one obtained by the O'Rourke method, since it is an exact method. The relative error is measured for PCA, PSO and DE. Since both PSO and DE present a stochastic component, 300 runs were performed for each algorithm and each data set, stopping after 200 iterations or when no improvement was made for 10 consecutive iterations; the best and worst results are presented, as well as the mean and variance of the relative error. The parameters were chosen as follows:

PSO:

$$\texttt{population\_size} = 20,\ c = 0.5,\ s = 0.5,$$
$$w(k) = 0.4 + 0.5\left(\frac{200-k}{200}\right)$$

DE:

$$\texttt{population\_size} = 20,\ F = 0.5,\ \mathrm{CR} = 0.3,$$

Tables 4.1 and 4.2 shows that the proposed algorithm runs much faster than the exhaustive O'Rourke method, without compromising too much the quality of the solutions. PCA, on the other hand, runs extremely fast but only produces rough approximations.
The experiment showed that PSO sometimes leads to situations where the particles prematurely converge to a local minimum. This explains the difference between the best and worst results and indicates that the proposed method needs improvements regarding its robustness. The same is observed for DE but in a lesser measure.

Nevertheless, it is worth noting that, in the experiments conducted here, no parameter adjustments were made. Furthermore, both PSO and DE present numerous variations in the literature that are much more sophisticated than the basic versions used here, notably regarding

the multimodal aspect of the objective function. The intent of these short experiments was mainly to demonstrate the feasibility of adapting heuristic algorithms on Riemannian manifolds.

Moreover, the techniques presented here are similar to the ones presented in [CGM11] in that they outperform classical optimization techniques (resorting to the use of derivatives). Such classical methods cannot cope with the multimodal aspect of the OBB problem and therefore converge quickly to local optima in the vicinity of the initial iterate.

4.7 Conclusion

This preliminary experiment indicates that, while being stochastic methods, PSO and DE show to be both efficient and relatively reliable to solve the OBB problem. Furthermore, the early results obtained in this work are encouraging the development of PSO, DE and other metaheuristics over different search spaces and manifolds.

These early results are also reinforcing the observations presented in [CGM11], where derivative-free methods are also suggested to cope with the multimodal and non-differentiable aspects of the OBB problem. The results presented in [CGM11] also outperform derivative-based methods.

Some extensions of the proposed algorithms can be considered for further work. Among the many variations of the PSO and DE algorithms existing in the literature, strategies dealing more efficiently with multimodal objectives could be valuable. Approaches involving self-adaptive coefficients may also add robustness to the proposed method. Finally, a hybridization of PSO/DE with a directional search method (compass search, MADS, ...) could be envisaged as proposed in [VV09].

data set	method	time (s)	relative error (%)				
			min	max	3rd quartile	mean	var
set1	O'Rourke	2.8×10^{0}	0	-	-	-	
17 nodes	PCA	1.0×10^{-4}	5.7×10^{1}	-	-	-	
	PSO	1.1×10^{-2}	0.0×10^{0}	1.5×10^{0}	2.5×10^{-12}	7.9×10^{-2}	2.5×10^{-1}
	DE	1.5×10^{-2}	0.0×10^{0}	2.7×10^{2}	5.6×10^{-13}	9.1×10^{-1}	1.5×10^{1}
set2	O'Rourke	1.5×10^{1}	0	-	-	-	
51 nodes	PCA	1.0×10^{-4}	1.0×10^{2}	-	-	-	
	PSO	2.1×10^{-2}	9.1×10^{-14}	1.7×10^{-1}	1.4×10^{-2}	1.3×10^{-2}	2.0×10^{-2}
	DE	5.7×10^{-2}	1.2×10^{-13}	1.6×10^{-2}	7.8×10^{-9}	1.7×10^{-3}	4.2×10^{-3}
set3	O'Rourke	1.1×10^{2}	0	-	-	-	
132 nodes	PCA	2.0×10^{-4}	2.8×10^{1}	-	-	-	
	PSO	1.3×10^{-2}	0.0×10^{0}	1.2×10^{0}	1.9×10^{-1}	1.8×10^{-1}	2.3×10^{-1}
	DE	5.4×10^{-3}	0.0×10^{0}	5.7×10^{-1}	1.2×10^{-1}	5.8×10^{-2}	1.3×10^{-1}
set4	O'Rourke	2.0×10^{3}	0	-	-	-	
666 nodes	PCA	4.0×10^{-4}	1.9×10^{1}	-	-	-	
	PSO	9.9×10^{-1}	4.7×10^{-14}	5.2×10^{0}	1.8×10^{-3}	1.0×10^{-1}	5.8×10^{-1}
	DE	7.4×10^{-2}	1.2×10^{-14}	2.6×10^{-1}	1.6×10^{-3}	1.8×10^{-3}	1.6×10^{-2}

Table 4.1: Comparison of the performance of the PSO and DE algorithms with PCA and O'Rourke (1/2).

data set	method	time (s)	relative error (%)				
			min	max	3rd quartile	mean	var
set5	O'Rourke	2.9×10^{3}	0	-	-	-	
688 nodes	PCA	3.0×10^{-4}	2.1×10^{1}	-	-	-	
	PSO	2.3×10^{-1}	7.1×10^{-14}	5.2×10^{0}	4.9×10^{-4}	9.1×10^{-2}	6.1×10^{-1}
	DE	1.1×10^{-2}	0.0×10^{0}	1.2×10^{-1}	1.4×10^{-9}	1.2×10^{-3}	1.0×10^{-2}
set6	O'Rourke	1.1×10^{4}	0	-	-	-	
1560 nodes	PCA	4.9×10^{-3}	8.4×10^{1}	-	-	-	
	PSO	1.5×10^{-2}	1.3×10^{-14}	5.8×10^{-1}	8.3×10^{-13}	1.3×10^{-2}	6.7×10^{-2}
	DE	1.6×10^{-2}	1.3×10^{-14}	4.2×10^{-1}	8.6×10^{-13}	2.4×10^{-3}	2.7×10^{-2}
set7	O'Rourke	2.6×10^{4}	0	-	-	-	
2394 nodes	PCA	6.0×10^{-4}	2.3×10^{1}	-	-	-	
	PSO	3.4×10^{-1}	9.4×10^{-14}	5.2×10^{0}	4.1×10^{-4}	1.2×10^{-1}	7.0×10^{-1}
	DE	1.0×10^{-1}	0.0×10^{0}	2.1×10^{-1}	1.6×10^{-9}	1.0×10^{-3}	1.3×10^{-2}
set8	O'Rourke	1.8×10^{5}	0	-	-	-	
6479 nodes	PCA	3.7×10^{-1}	1.1×10^{2}	-	-	-	
	PSO	1.5×10^{-1}	8.6×10^{-13}	5.7×10^{-1}	1.2×10^{-3}	1.4×10^{-2}	6.9×10^{-2}
	DE	6.3×10^{-2}	1.5×10^{-12}	1.5×10^{-1}	1.1×10^{-10}	9.4×10^{-4}	1.1×10^{-2}

Table 4.2: Comparison of the performance of the PSO and DE algorithms with PCA and O'Rourke (2/2).

Chapter 5

Low multilinear rank approximation of higher-order tensors

Chapter overview
In this second applicative chapter, we are interested in finding the best low multilinear rank approximation of a given tensor. The multilinear rank of a tensor is one of the possible generalizations for the concept of matrix rank. This problem has been formulated as an optimization problem over the Grassmann manifold [Ish09] and it has been shown that the objective function presents multiple minima [IAVD10]. In order to investigate the landscape of this cost function, we propose an adaptation of the Particle Swarm Optimization algorithm (PSO) to the Cartesian product of Grassmann manifolds. The Guaranteed Convergence PSO, proposed by van den Bergh in [vdBE02], is modified, including a gradient component, so as to search for optimal solutions over this search space. The operations involved in the PSO algorithm are redefined using concepts of differential geometry. We present some preliminary numerical experiments and we discuss the ability of the proposed method to address the multimodal aspects of the studied problem.

The material of this chapter is based on the following publication:
[BIA10] P.B. Borckmans, M. Ishteva, and P.-A. Absil. A Modified Particle Swarm Optimization Algorithm for the Best Low Multilinear Rank Approximation of Higher-Order Tensors. *In Proceedings of the 7th international conference on Swarm intelligence*, ANTS10, pages 1323, Berlin, Heidelberg, 2010. Springer-Verlag.

5.1 Introduction

Although the generalization of matrices to higher-order tensors is quite natural, the concept of matrix rank cannot be extended in a straightforward way. However, there are some propositions to define the rank for tensors, in an effort to maintain some properties found in the matrix theory. A particular choice, explained in section 5.2.1, is the multilinear rank, which is a direct generalization of the row and column ranks for matrices [Hit27a, Hit27b].

The problem we investigate in this chapter is to find the best low multilinear rank approximation of a given tensor. This problem, presented in section 5.2.2, is a natural extension of the low rank matrix approximation problem. Amongst the many motivations behind these problems, one can cite data compression, dimensionality reduction and independent component analysis (ICA) in domains as various as statistics [McC87], signal processing [ABB$^+$07] and image processing [VT03].

In the matrix case, it is known that the best approximation is given by the truncated singular value decomposition (SVD). A generalization of this concept for higher-order tensors is called the higher-order SVD (HOSVD) [DDV00a], or more generally Tucker decomposition [Tuc64, Tuc66]. However, taking truncated versions of these decompositions no longer leads to optimal approximations for tensors. That being said, the search for low rank tensor approximation can be formulated as an optimization problem on a particular search space, namely the Grassmann manifold, as explained in section 5.2.4.

This optimization problem has been studied before, most notably in [DDV00b, Kd80, Kro08], and has been shown to present multiple local minima. In [IAVD10], the authors show that there can be many of these local minima but that the differences in terms of the cost function can be very small. What really differs are the subspaces spanned by the projection matrices corresponding to different solutions. These

differences may have important consequences for applications and motivate the need to investigate the landscape of the objective function, without getting trapped in the first encountered local minimum. In order to perform this investigation, one needs to design global optimization methods on the Grassmann manifold. Some methods dealing with such search spaces are available in the literature. The global aspect of these methods rely either on a stochastic component or on a population framework. In [LSG04] for example, the authors propose a stochastic gradient algorithm. In [Dre07] Dreisigmeyer defines a canvas to develop direct-search optimization method on Riemannian manifolds, such as the Nelder-Mead algorithm and the lower-triangular mesh adaptive direct search (LTMADS) algorithm [ACD07].

This chapter presents preliminary results towards applying swarm intelligence techniques to efficiently explore the landscape of the objective function appearing in the low multilinear rank approximation problem. In this proof-of-concept chapter, we concentrate on the most basic Particle Swarm Optimization (PSO) algorithms, extend them to tackle the tensor low rank approximation cost function defined on a Cartesian product of Grassmann manifolds, and report on numerical experiments that illustrate the applicability of the obtained algorithms. More specifically, we start from the Guaranteed Convergence PSO (GCPSO), as presented by van den Bergh *et al.* in [vdBE02], and modify it so as to search for optimal solutions over the Grassmann manifold. A gradient component is also included in the method to improve its convergence rate.

The chapter is organized as follows. Section 5.2 reviews the best low multilinear rank approximation of tensors, its formulation on the Grassmann manifold, and the issue of local minima. Section 5.3 presents how GCPSO is adapted to the Grassmann manifold, using concepts of differential geometry, and introduces a gradient component in the algorithm. Finally, section 5.4 presents numerical results and discusses the ability of the proposed method to deal with the multimodal aspects of the best low multilinear rank approximation problem.

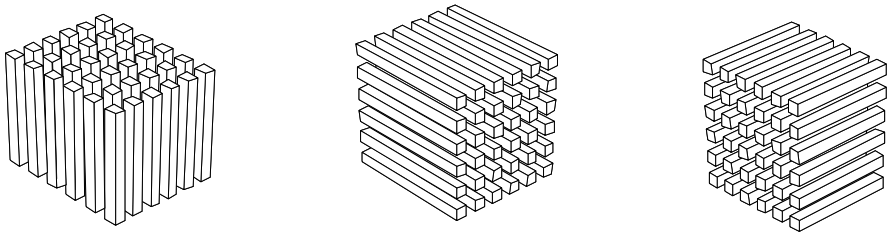


Figure 5.1: Illustration of the mode-n vectors (or fibers) of a $(5 \times 5 \times 5)$-tensor.

5.2 Low Multilinear Rank Approximation Problem

5.2.1 Tensor generalities

Before diving into the problem we want to address, we give here some general concepts and definitions about tensors that will be necessary in the following. An *Nth-order real tensor* $\mathscr{A}$ is a N-way array of real numbers; a first-order tensor is thus a vector, a second-order tensor is a matrix, a third-order tensor is a 3D-array. In the following, we will focus on third-order tensors, for clarity, but extension to higher-order tensors is mainly technical. The *mode-n rank* of $\mathscr{A}$, $n = 1, \ldots, N$, denoted by $\text{rank}_n(\mathscr{A})$, is the number of linearly independent *mode-n* vectors, also called fibers represented in Figure 5.1, *i.e.* the vectors obtained by varying the n-th index of $\mathscr{A}$ while fixing the others. This concept is an extension of the row and column ranks of a matrix, with the important difference that the different mode-n ranks are generally not the same. The *multilinear rank* of an Nth-order tensor $\mathscr{A}$ is then the N-tuple of its *mode-n* ranks.

The *matrix representations* $\boldsymbol{A}_{(n)}, n = 1, 2, 3$, of a third-order tensor $\mathscr{A}$, also called *flattenings* or *unfoldings* of $\mathscr{A}$, are obtained by concatenating the *mode-n* vectors of $\mathscr{A}$ in a specific order, given by:

$$\left(\boldsymbol{A}_{(1)}\right)_{i_1,(i_2-1)I_3+i_3} = \left(\boldsymbol{A}_{(2)}\right)_{i_2,(i_3-1)I_1+i_1} = \left(\boldsymbol{A}_{(3)}\right)_{i_3,(i_1-1)I_2+i_2} = a_{i_1 i_2 i_3}, \tag{5.1}$$

where $a_{i_1 i_2 i_3}$ denotes the scalar entry of the $\mathscr{A}$ at the position (row, column and "slice" indices) (i_1, i_2, i_3). Note that it results from this definition that $\text{rank}_n(\mathscr{A}) = rank(\boldsymbol{A}_{(n)})$.

In the following we will use the notation $\|\mathscr{A}\|$ to denote the *Frobenius*

norm of the tensor $\mathscr{A}$:

$$\|\mathscr{A}\| = \sqrt{\langle \mathscr{A}, \mathscr{A} \rangle},$$

where $\langle \mathscr{A}, \mathscr{B} \rangle$ denotes the *scalar product* of two tensors $\mathscr{A}$ and $\mathscr{B}$:

$$\langle \mathscr{A}, \mathscr{B} \rangle = \sum_{i_1} \sum_{i_2} \sum_{i_3} a_{i_1 i_2 i_3} b_{i_1 i_2 i_3}.$$

5.2.2 Problem statement

Finding the best low multilinear rank of a tensor $\mathscr{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ consists in finding another tensor $\hat{\mathscr{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ with mode-n ranks bounded by pre-specified R_1, R_2, R_3 respectively, with $R_i \leq I_i$, that is as close as possible to $\mathscr{A}$. This can be expressed by the following optimization problem:

$$\begin{aligned} &\min_{\hat{\mathscr{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}} \|\mathscr{A} - \hat{\mathscr{A}}\|^2 \\ &\text{s.t.} \qquad rank_1(\hat{\mathscr{A}}) \leq R_1,\ rank_2(\hat{\mathscr{A}}) \leq R_2,\ rank_3(\hat{\mathscr{A}}) \leq R_3. \end{aligned} \tag{5.2}$$

It has been shown in [DDV00b, Kd80, Kro08] that this problem is equivalent to the problem of maximizing the function

$$\begin{aligned} \bar{g}: \ & \mathrm{St}(R_1, I_1) \times \mathrm{St}(R_2, I_2) \times \mathrm{St}(R_3, I_3) \to \mathbb{R}, \\ & (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) \mapsto \|\boldsymbol{U}^\top \boldsymbol{A}_{(1)} (\boldsymbol{V} \otimes \boldsymbol{W})\|^2 \end{aligned} \tag{5.3}$$

where $\mathrm{St}(p, n)$ stands for the set of column-wise orthonormal $(n \times p)$-matrices, *i.e.* the Stiefel manifold, and $\otimes$ is the Kronecker product.

An interesting property of the cost function $\bar{g}$ is the following:

$$\|\boldsymbol{U}^\top \boldsymbol{A}_{(1)} (\boldsymbol{V} \otimes \boldsymbol{W})\|^2 = \|\boldsymbol{V}^\top \boldsymbol{A}_{(2)} (\boldsymbol{W} \otimes \boldsymbol{U})\|^2 = \|\boldsymbol{W}^\top \boldsymbol{A}_{(3)} (\boldsymbol{U} \otimes \boldsymbol{V})\|^2.$$

This property allows one to optimize the function $\bar{g}$ over one of the three variables $\boldsymbol{U}, \boldsymbol{V}$ or $\boldsymbol{W}$ while keeping the other two fixed.

This function $\bar{g}$ can be seen as a natural generalization of the matrix case where the best rank-R approximation $\hat{\boldsymbol{A}}$ of a matrix $\boldsymbol{A} \in \mathbb{R}^{I_1 \times I_2}$ is given by maximizing $\|\boldsymbol{U}^\top \boldsymbol{A} \boldsymbol{V}\|$, with $\boldsymbol{U} \in \mathbb{R}^{I_1 \times R}$ and $\boldsymbol{V} \in \mathbb{R}^{I_2 \times R}$ being column-wise orthogonal matrices (see [DV04]). In fact, in this case, the

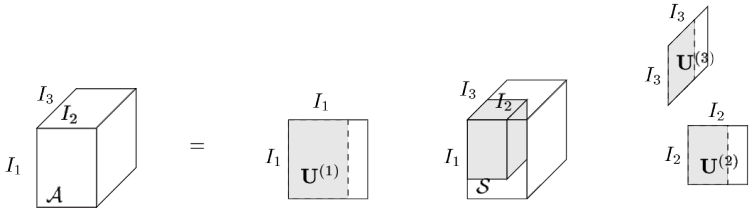


Figure 5.2: Illustration of the HOSVD of a 3rd-order tensor $\mathscr{A}$ of dimensions $I_1 \times I_2 \times I_3$ (illustration reproduced from [Ish09]).

best solution is known to be given by the truncated SVD. For higher-order tensors, this notion can be extended to HOSVD (illustrated in Figure 5.2), but it no longer leads to optimal solutions. Anyhow, the truncated HOSVD is often used as a good starting point for optimization algorithms, even though it has been shown that this does not always imply convergence to the global optimum [IAVD10].

A noteworthy remark about this approximation problem is that it belongs to the class of tensor problems that are likely to be NP-hard as investigated in Heng Lim's work (Most tensor problems are NP-hard, [HL09]).

5.2.3 Local minima

In [IAVD10], the authors show that the best low multilinear rank approximation problem (5.2) presents local non-global minima. This is a major difference with the matrix case where the problem is uni-modal. The authors considered tensors with low multilinear rank, perturbed by a small amount of additive noise. For such test tensors, the local algorithms proposed in [Ish09] converge to a small amount of different minima. After increasing the noise level, the tensors become less structured and more local minima are found. This behavior is related to the distribution of the mode-n singular values (the singular values of the matrices $A_{(n)}$). When the noise level is low, there is a gap between the singular values, whereas when the noise level increases, the gap becomes small or non-existent. In this case, the best low multilinear rank approximation becomes a difficult problem since we are looking for a structure that is not present: there are many equally good, or equally bad, solutions.

Concerning the values of the cost function at the different local min-

ima, they seem to be similar [IAVD10]. This means that in applications where the multilinear rank approximation is merely used as a compression tool for memory savings, using a local optimizer such as the ones proposed in [Ish09] is efficient and reliable.

On the other hand, the subspaces spanned by two matrices U_1 and U_2 corresponding to two different local optima in (5.3) are very different and the same holds for V and W [IAVD10]. In applications where these subspaces are important, such as dimensionality reduction, this observation may be important to consider. A last remark made in [IAVD10] is that the global optimum is not necessarily the desired one. The authors show that when a tensor with low multilinear rank is affected by noise, the subspaces corresponding to the global optimum are not always the closest to the subspaces corresponding to the original noise-free tensor, especially for high noise levels.

All of these remarks motivate the need to develop methods that are able to investigate the landscape of the objective function (5.3) in a global perspective. In [Ish09], the author suggests that local algorithms can be run several times, from different initial conditions, in order to discover the different local optima. In this chapter, we propose a more sophisticated, population-based method, in the perspective that one run of such a global method with a given number of individuals would discover the different optima more efficiently than the same amount of runs of a local optimizer, thanks to collaboration between individuals.

5.2.4 The Grassmann manifold $\mathrm{Gr}(p, n)$

The cost function $\bar{g}$ defined in (5.3) has an important invariance property. For any orthogonal matrices $\boldsymbol{Q}^{(i)} \in O_{R_i}, i = 1, 2, 3$, one can show that:

$$\bar{g}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}) = \bar{g}(\boldsymbol{U}\boldsymbol{Q}^{(1)}, \boldsymbol{V}\boldsymbol{Q}^{(2)}, \boldsymbol{W}\boldsymbol{Q}^{(3)}). \tag{5.4}$$

This means that we are not looking for the exact elements of $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{W}$, but for the subspaces that their columns span. Consequently, the optimal solutions of $\bar{g}$ are not isolated since for every triplet $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})$, there is a whole equivalence class of triplets leading to the same cost. For PSO methods, this is a source of difficulty since particles may converge to minima that are structurally identical (they yield the same cost) while being far apart from each other in the $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W})$ space. One way of getting rid of this redundancy is to work conceptually on a smaller-dimensional search space by making use of the Grassmann manifold.

The Grassmann manifold $G(p,n)$ is the set of all p-planes in $\mathbb{R}^n$. It can be thought of as the quotient manifold $\mathcal{M} = \mathrm{St}(p,n)/O_p$ (see e.g. [EAS98]). A point in $G(p,n)$ thus represents a whole equivalence class of matrices spanning the same p-dimensional subspace. In practice, we will represent a point in $G(p,n)$ by choosing an arbitrary member in $\mathrm{St}(p,n)$ of its equivalence class.

5.3 Particle Swarm Optimization

In this section, we develop an optimization algorithm based on PSO for the best low multilinear rank approximation problem. In order to do so, we need to adapt PSO to the cost function (5.3), defined on a Cartesian product of three Grassmann manifolds. This section recalls the basic PSO and GCPSO algorithm formulations, then presents the adaptation of their different components to the Grassmann manifold. Finally the algorithm for the Cartesian product is detailed.

5.3.1 PSO in $\mathbb{R}^n$

First introduced in 1995 by Eberhart and Kennedy [KE95], PSO is a stochastic population-based algorithm. Particles are points evolving in the search space, following simple rules, mimicking the behaviour of social groups. Points are initialized randomly in $\mathbb{R}^n$ and the driving force of the optimization process is given by the following update equations (iterated over k), for each particle (indexed by i):

$$\begin{aligned} \boldsymbol{v}_i(k+1) &= \underbrace{w(k)\boldsymbol{v}_i(k)}_{inertia} + \underbrace{c\alpha_i(k)\left(\boldsymbol{y}_i(k) - \boldsymbol{x}_i(k)\right)}_{nostalgia} \\ &\quad + \underbrace{s\beta_i(k)\left(\hat{\boldsymbol{y}}(k) - \boldsymbol{x}_i(k)\right)}_{social} \qquad (5.5) \\ \boldsymbol{x}_i(k+1) &= \boldsymbol{x}_i(k) + \boldsymbol{v}_i(k+1)\,, \qquad (5.6) \end{aligned}$$

where $\boldsymbol{x}_i$ denotes position, $\boldsymbol{v}_i$ denotes velocity, $\boldsymbol{y}_i$ is the personal best position so far, $\hat{\boldsymbol{y}}$ is the global best position of the swarm so far; w is inertia coefficient (usually dynamic), c and s are adjustable coefficients, and α and β are random components. As can be seen, the behaviour of each particle is dictated by velocity increments composed of three simple components: inertia, cognitition (nostalgic behaviour) and social behaviour. At the end of each iteration, y_i and $\hat{y}$ are updated.

5.3.2 GCPSO and gradient

The basic PSO algorithm introduced in the previous section does not present any guarantee about the convergence to a minimum, not even a local one. Actually, the only thing that is guaranteed is that each particle converges to a point on the line joining its best position and the global best position of the whole swarm so far. In order to ensure convergence to a stationary point, van den Bergh *et al.* proposed in [vdBE02] a slight variation of the original PSO, called Guaranteed Convergence PSO, avoiding stagnation of the particles. In this model, all the particles follow the velocity update equation (5.5) except for the best particle at current iteration k, denoted by the index $i = \tau$, which follows the velocity equation

$$\boldsymbol{v}_\tau(k+1) = \underbrace{\hat{\boldsymbol{y}}(k) - \boldsymbol{x}_\tau(k)}_{\text{reset}} + \underbrace{w(k)\boldsymbol{v}_\tau(k)}_{\text{inertia}} + \underbrace{\rho(k)(1 - 2\boldsymbol{\gamma}(k))}_{\text{sampling}}\,, \tag{5.7}$$

where γ is a random vector uniformly distributed over $[0, 1]$. This last equation forces the best particle to randomly reset its position in a ball around the current best position. The radius of this ball, $\rho(k)$, is increased when a better position is sampled and it is reduced when the sampling fails, therefore ensuring convergence to a local minimum. For more in-depth information about PSO, GCPSO and other variants, the reader is referred to [Eng06].

When the gradient of the objective function is available, the sampling term in (5.7) can be advantageously replaced by a term involving the gradient:

$$\boldsymbol{v}_\tau(k+1) = \underbrace{\hat{\boldsymbol{y}}(k) - \boldsymbol{x}_\tau(k)}_{\text{reset}} + \underbrace{w(k)\boldsymbol{v}_\tau(k)}_{\text{inertia}} - \underbrace{\delta(k)\nabla f(\boldsymbol{x}_\tau(k))}_{\text{descent}}\,, \tag{5.8}$$

where $\delta(k)$ controls how much use of the gradient is made over time. The idea there is that the whole swarm is moving through the search space using mutual interactions, while the best individual refines its position in the best way that is locally possible, *i.e.* along its steepest descent direction. This strategy has the advantage of preserving the exploration quality of PSO while improving the exploitation of potential optima that are found along the way.

5.3.3 Adapting GCPSO to the Grassmann manifold

In its original form, PSO is described for particles in $\mathbb{R}^n$: $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$ and the operations involved in the update equations $(+,-,.)$ are the usual vectorial addition, subtraction and scaling. In order to adapt standard PSO to the Grassmannian search space, one must redefine x, v and the operations mentioned above.

A point $\boldsymbol{x}$ on the Grassmann manifold $G(p,n)$ is represented by a matrix $\boldsymbol{X} \in \mathrm{St}(p,n)$ and likewise for $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. The velocity $\boldsymbol{v}$ is now an element of $\mathrm{T}_{\boldsymbol{x}}G(p,n)$, the tangent space to $G(p,n)$ centred at $\boldsymbol{x}$, represented by a matrix in $\mathcal{H}_{\mathbf{X}} := \left\{ \dot{\boldsymbol{X}} \in \mathbb{R}^{n\times p} : \boldsymbol{X}^\top \dot{\boldsymbol{X}} = 0 \right\}$; see [AMS08, §3.6.2] for details. In the following, we will abuse notation and write $\boldsymbol{X} \in G(p,n)$ for an element of $\mathrm{St}(p,n)$ meant to represent an element of $G(p,n)$, and $\dot{\boldsymbol{X}} \in \mathrm{T}_{\boldsymbol{X}}G(p,n)$ for an element of $\mathcal{H}_{\mathbf{X}}$ meant to represent an element of $\mathrm{T}_{\boldsymbol{X}}G(p,n)$.

In order to rewrite the velocity and position update equations (5.5)–(5.6), one needs to be able to compute a geodesic starting at a given point with a given initial velocity, and to compute the initial velocity of a geodesic linking two points. The exponential map and the logarithmic map are the respective tools to address these questions (see e.g. [AMS04]). We consider the geodesics induced by the (essentially) unique rotation-invariant metric on $G(p,n)$.

The geodesic starting at a point $\boldsymbol{X} \in G(p,n)$ with an initial velocity $\dot{\boldsymbol{X}} \in \mathrm{T}_{\boldsymbol{X}}G(p,n)$ is given by (see [AMS04, §3.7]):

$$\boldsymbol{\gamma}(t) = \mathrm{Exp}_{\boldsymbol{X}}(t\dot{\boldsymbol{X}}) := \boldsymbol{X}\boldsymbol{V}\cos(t\boldsymbol{\Theta}) + \boldsymbol{U}\sin(t\boldsymbol{\Theta}) \;, \tag{5.9}$$

where $\dot{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Theta}\boldsymbol{V}^\top$ is a thin-SVD.

The initial velocity $\dot{\boldsymbol{X}} \in \mathrm{T}_{\boldsymbol{X}}G(p,n)$ of the geodesic linking two points $\boldsymbol{X}$ and $\boldsymbol{Y}$ on $G(p,n)$ is given by (see [AMS04, §3.8]):

$$\dot{\boldsymbol{X}} = \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) := \boldsymbol{U}\mathrm{atan}(\boldsymbol{\Sigma})\boldsymbol{V}^\top \;, \tag{5.10}$$

where $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ are given by the thin-SVD of $\Pi_{\boldsymbol{X}}^{\perp}\boldsymbol{Y}(\boldsymbol{X}^\top\boldsymbol{Y})^{-1}$ (where $\Pi_{\boldsymbol{X}}^{\perp} := \left(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^\top\right)$ is the orthogonal projection onto $\mathcal{H}_{\mathbf{X}}$):

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \Pi_{\boldsymbol{X}}^{\perp}\boldsymbol{Y}\left(\boldsymbol{X}^\top\boldsymbol{Y}\right)^{-1} \;. \tag{5.11}$$

Using relations (5.9) and (5.10), the velocity and position update

equations (5.5)–(5.6) can now be rewritten as follows:

$$\dot{\boldsymbol{X}}_i(k+1) = w(k)\frac{\mathrm{d}}{\mathrm{dt}}\,\mathrm{Exp}_{\boldsymbol{X}_i(k-1)}\left(t\dot{\boldsymbol{X}}_i(k)\right)\Big|_{t=1} + c\alpha_i(k)\mathrm{Log}_{\boldsymbol{X}_i(k)}\boldsymbol{Y}_i(k) + s\beta_i(k)\mathrm{Log}_{\boldsymbol{X}_i(k)}\hat{\boldsymbol{Y}}_i(k) \quad (5.12)$$

$$\boldsymbol{X}_i(k+1) = \mathrm{Exp}_{\boldsymbol{X}_i(k)}\dot{\boldsymbol{X}}_i(k+1) \quad (5.13)$$

Equation (5.7) for GCPSO involves 3 terms. The two first terms are similar to the ones in equation (5.5) and can therefore be adapted in the same way. To adapt the last term (sampling), one needs to be able to generate a displacement in $\mathrm{T}_{\boldsymbol{X}}G(p,n)$, the tangent space at $\boldsymbol{X}$, so as to generate a random matrix $\boldsymbol{Y} \in G(p,n)$ in a ball of radius ρ around $X \in G(p,n)$. The displacement $\boldsymbol{Z}$ can be computed as $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ where $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ are chosen as follows: draw $\boldsymbol{R} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{W} \in \mathbb{R}^{p\times p}$ from the normal distribution, $(s_1, \ldots, s_p)$ from the uniform distribution on $[0,1]$ and compute:

$$\begin{array}{ccc} \boldsymbol{U} = \mathrm{qf}(\boldsymbol{Q}) & & \\ \boldsymbol{Q} = \Pi^{\perp}_{\boldsymbol{X}}(\boldsymbol{R}) & \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_p \end{bmatrix} & \boldsymbol{V} = \mathrm{qf}(\boldsymbol{W}) \\ & \sigma_i = \rho(k)(1-2s_i) & \end{array} \quad (5.14)$$

where qf returns the Q-factor of the QR decomposition of its matrix argument.

Equation (5.7) can thus finally be rewritten as follows:

$$\dot{\boldsymbol{X}}_\tau(k+1) = \mathrm{Log}_{\boldsymbol{X}_\tau(k)}\hat{\boldsymbol{Y}}(k) + w(k)\frac{\mathrm{d}}{\mathrm{dt}}\,\mathrm{Exp}_{\boldsymbol{X}_\tau(k-1)}\left(t\dot{\boldsymbol{X}}_\tau(k)\right)\Big|_{t=1} + \boldsymbol{U}\boldsymbol{\Sigma}_{\rho(k)}\boldsymbol{V}^\top \quad (5.15)$$

Finally, equation (5.8) can be rewritten in the following way:

$$\dot{\boldsymbol{X}}_\tau(k+1) = \mathrm{Log}_{\boldsymbol{X}_\tau(k)}\hat{\boldsymbol{Y}}(k) + w(k)\frac{\mathrm{d}}{\mathrm{dt}}\,\mathrm{Exp}_{\boldsymbol{X}_\tau(k-1)}\left(t\dot{\boldsymbol{X}}_\tau(k)\right)\Big|_{t=1} - \delta(k)\,\mathrm{grad}\,f(\boldsymbol{X}_\tau(k))\,, \quad (5.16)$$

5.3.4 PSO in $\mathrm{Gr}(R_1, I_1) \times \mathrm{Gr}(R_2, I_2) \times \mathrm{Gr}(R_3, I_3)$

The previous section showed how PSO and GCPSO can be adapted to the Grassmann manifold. For the purpose of optimizing $\bar{g}$ (5.3), the search space is actually a Cartesian product of three Grassmann manifolds, $\mathrm{Gr}(R_1, I_1) \times \mathrm{Gr}(R_2, I_2) \times \mathrm{Gr}(R_3, I_3)$ and we endow this search space with the standard product metric.

The particles are thus elements described as follows:

$$\{(U, V, W) : U \in \mathrm{Gr}(R_1, I_1), V \in \mathrm{Gr}(R_2, I_2), W \in \mathrm{Gr}(R_3, I_3)\}.$$

The formulas for the exponential and the logarithm generalize component-wise as follows:

$$\mathrm{Exp}_{(\boldsymbol{U},\boldsymbol{V},\boldsymbol{W})}\left(\dot{\boldsymbol{U}}, \dot{\boldsymbol{V}}, \dot{\boldsymbol{W}}\right) = \left(\mathrm{Exp}_{\boldsymbol{U}}\dot{\boldsymbol{U}}, \mathrm{Exp}_{\boldsymbol{V}}\dot{\boldsymbol{V}}, \mathrm{Exp}_{\boldsymbol{W}}\dot{\boldsymbol{W}}\right) \quad (5.17)$$

$$\mathrm{Log}_{(\boldsymbol{U}_a,\boldsymbol{V}_a,\boldsymbol{W}_a)}\left(\boldsymbol{U}_b, \boldsymbol{V}_b, \boldsymbol{W}_b\right) = \left(\mathrm{Log}_{\boldsymbol{U}_a}\boldsymbol{U}_b, \mathrm{Log}_{\boldsymbol{V}_a}\boldsymbol{V}_b, \mathrm{Log}_{\boldsymbol{W}_a}\boldsymbol{W}_b\right) \quad (5.18)$$

Equations (5.12)-(5.16) can now be adapted to the search space using (5.17)-(5.18).

Using equations (5.12)-(5.18), we finally propose Algorithm 5.1.

5.4 Numerical results

In this section, we present some preliminary numerical experiments that illustrate the ability of the proposed algorithm to find the global optimum of the cost function $\bar{g}$. We build a set of test tensors as follows:

- we choose dimensions (I_1, I_2, I_3), we fix the multilinear rank $R = (R_1, R_2, R_3)$ and noise levels σ_i
- we build a set of tensors $\mathscr{A}_i$ with multilinear rank R, perturbed with additive noise controlled by σ_i, using

$$\tilde{\mathscr{A}_i} = \mathscr{T}_i/\|\mathscr{T}_i\| + \sigma_i * \mathscr{E}_i/\|\mathscr{E}_i\|, \quad \mathscr{A}_i = \tilde{\mathscr{A}_i}/\|\tilde{\mathscr{A}_i}\|, \quad (5.19)$$

  where $\mathscr{T}_i$ is obtained as a product of a (R_1, R_2, R_3)-tensor $\mathscr{B}_i$ and three column-wise orthonormal matrices with matching dimensions (to reach an (I_1, I_2, I_3)-tensor). The elements of $\mathscr{B}_i$, $\mathscr{E}_i$ and the three matrices are drawn from a normal distribution with zero mean and unit standard deviation. The Q-factor of the QR decomposition is then taken for the matrices.

Algorithm 5.1 GCPSO with gradient adapted to the best low multilinear rank tensor approximation problem.

Output: $\boldsymbol{p} \in \Omega$, an admissible point

Draw initial points $(\boldsymbol{U}_i(0), \boldsymbol{V}_i(0), \boldsymbol{W}_i(0))$, $\forall i$, from the normal distribution and taking the Q-factor

Set null initial velocities $(\dot{\boldsymbol{U}}_i(0), \dot{\boldsymbol{V}}_i(0), \dot{\boldsymbol{W}}_i(0))$

while $k < max\_iter$ **do**

Update personal bests $\left(\boldsymbol{Y}^{\boldsymbol{U}_i(k)}, \boldsymbol{Y}^{\boldsymbol{V}_i(k)}, \boldsymbol{Y}^{\boldsymbol{W}_i(k)}\right)$, $\forall i$

Update global bests $\left(\hat{\boldsymbol{Y}}^{\boldsymbol{U}(k)}, \hat{\boldsymbol{Y}}^{\boldsymbol{V}(k)}, \hat{\boldsymbol{Y}}^{\boldsymbol{W}(k)}\right)$ and τ

$\forall i \neq \tau$, compute $\left(\dot{\boldsymbol{U}}_i(k), \dot{\boldsymbol{V}}_i(k), \dot{\boldsymbol{W}}_i(k)\right)$ using (5.12)

if $k \leq iter\_GCPSO$ **then**

Compute $\left(\dot{\boldsymbol{U}}_\tau(k), \dot{\boldsymbol{V}}_\tau(k), \dot{\boldsymbol{W}}_\tau(k)\right)$ using (5.15) *(sampling)*

else

Compute $\left(\dot{\boldsymbol{U}}_\tau(k), \dot{\boldsymbol{V}}_\tau(k), \dot{\boldsymbol{W}}_\tau(k)\right)$ using (5.16) *(gradient)*

end if

Compute $(\boldsymbol{U}_i(k), \boldsymbol{V}_i(k), \boldsymbol{W}_i(k))$ using (5.13)

Update $\rho(k)$ or $\delta(k)$

end while

return $\bar{g}\left(\boldsymbol{U}_\tau(k), \boldsymbol{V}_\tau(k), \boldsymbol{W}_\tau(k)\right)$

For each tensor of this set, we perform 100 runs of a local optimizer, namely the higher order orthogonal iteration (HOOI) [DDV00b, Kro08], with randomly-selected initial conditions, in order to get a putative global optimum $(\boldsymbol{U}_*, \boldsymbol{V}_*, \boldsymbol{W}_*)$. Then, we run the proposed algorithm 5.1, and we declare success if the best objective value $\bar{g}_\star$ found by the algorithm falls within $\epsilon = 10^{-8}$ of $\bar{g}(U_*, V_*, W_*)$.

The parameters for PSO were chosen as follows:

- `population_size` $= 10$, $c = 0.5$, $s = 0.5$, `max_iter` $= 1000$, `iter_GCPSO` $= 400$;
- $w(k)$ linearly decreasing from 0.9 to 0.4;
- $\rho(k)$ doubling when 3 consecutive successes, halving when 5 consecutive failures;
- $\delta(k)$ linearly decreasing from 1 to 0.

The following table shows the rate of success (a rate of 1 means that success was always observed) for the cases $(I_1, I_2, I_3) = (10, 10, 10)$, $(R_1, R_2, R_3) = (2, 2, 2)$ and $(I_1, I_2, I_3) = (10, 12, 15)$, $(R_1, R_2, R_3) = (2, 3, 4)$, evaluated over 10 runs of algorithm 5.1.

	σ	1	2	3	5	10
case 1	success rate	1	1	.9	.4	0
case 2	success rate	1	1	.8	.3	.1

We see that algorithm 5.1 is very successful when the noise level is low, even though local non-global optima are already present in the objective function. We also see that when the noise level becomes very high, the landscape of the objective function becomes so intricate that the performance of the proposed algorithm degrades.

5.5 Conclusion

We have introduced an adaptation of the GCPSO algorithm, including a gradient component, for the best low multilinear rank approximation problem. The proposed algorithm shows capacities to discover global optima, often without getting trapped in suboptimal solutions.

In future work, we will explore the niching capabilities of the PSO algorithm, as presented *e.g.* in [ZZL05, BEvdB02]. These variations

improve the ability of PSO to discover and maintain several local minima and could be used in this application to investigate the landscape of the objective function in a more informative way.

Further investigation of the problem might also involve other optimization techniques such as evolutionary algorithms and memetic algorithms. A comparison with such existing population-based algorithms might give more insight about the difficulty of the considered problem.

Chapter 6

Independent component analysis using the range contrast function

Chapter overview
It is seemingly paradoxical to the classical definition of the independent component analysis (ICA), that in reality the true sources are often not strictly uncorrelated. With this in mind, this chapter concerns a framework to extract quasi-uncorrelated sources with finite supports by optimizing a range-based contrast function under unit-norm constraints (to handle the inherent scaling indeterminacy of ICA) but without orthogonality constraints. Albeit the appealing contrast properties of the range-based function, e.g., the absence of mixing local optima, the function is not differentiable everywhere. Unfortunately, there is a dearth of literature on derivative-free optimization methods that effectively handle such a nonsmooth yet promising contrast function. This is the compelling reason for the design of a nonsmooth optimization algorithm on a manifold of matrices having unit-norm columns with the following objectives: (i) to ascertain convergence to a Clarke stationary point of the contrast function; (ii) to adhere to the necessary unit-norm constraints more naturally. The proposed nonsmooth optimization algorithm crucially relies on the design and analysis of an extension of the Mesh Adaptive Direct Search (MADS) method to handle locally Lipschitz objective functions defined on the sphere. The applicability of the algorithm in the ICA domain is demonstrated with simulations involving natural, face, aerial and texture images.

This chapter closely corresponds to the following recently-submitted collaborative work:
[BCA13] S. Easter Selvan, Pierre B. Borckmans, Amit Chattopadhyay, and P.-A. Absil. Spherical Mesh Adaptive Direct Search for Separating Quasi-uncorrelated Sources by Range-Based Independent Component Analysis. *Neural Comput.*, 2013, accepted.

6.1 Introduction

Independent component analysis (ICA) is a blind source separation technique that attempts to linearly recombine given signals into maximally statistically independent components. The task is usually formulated as minimizing[1] an objective function, termed *contrast function*, that quantifies the "level of dependence" of a collection of random variables. In practical applications, one is often given an $n \times T$ matrix $\mathbf{M}$ containing T samples of n real-valued signals, and the ICA task is then to find an unmixing matrix $\mathbf{X}$ such that the matrix $\mathbf{X}^\top\mathbf{M}$ looks as much as possible, in the sense of a finite-sample contrast function, like T observations of n statistically independent random variables. ICA is a very active research topic—see, e.g., the surveys [Hyv99b, Hyv12]—with many real-world applications, such as biomedical signal processing, remote sensing, seismic signal analysis, denoising in electric and magnetic circuits, and medical image analysis.

A range-based contrast function for the simultaneous extraction of bounded sources was first introduced in [PI96] and further investigated in [LVV08,LVV06,Vri07]. A finite-sample estimate of this contrast, with low sensitivity to noise and outliers, can be obtained (see Section 6.6 for details) using the finite-sample estimator of the range using order statistics, as proposed in [VLV07, §V]. We adopt this contrast in this work due to the following reasons. (i) It is endowed with the discriminacy property, i.e., it is devoid of mixing local optima [PV06]. This means that any local minimum of the contrast function corresponds to an unmixing matrix estimate. (ii) It is suitable for estimating the sources involving signals/images which are in general bounded. (iii) It has been proved to

[1] As in [PV06] and in keeping with the recent optimization literature, we adopt the convention that the contrast function is to be minimized. The literature that adopts the opposite convention is encompassed by taking the opposite of the contrast function considered.

be suitable for large-scale, ill-conditioned, and noisy mixtures [LVV08]. (iv) The finite-sample range-based contrast function is less expensive to evaluate than classical contrast functions such as the Shannon-entropy-based mutual information (MI).

However, minimizing this range-based contrast function presents two difficulties. First, in connection with the scaling indeterminacy of the ICA task, it is common to impose a *unit-norm constraint* on the columns of the candidate unmixing matrix $\mathbf{X}$; see, e.g., [VLV07, §II] and [AG06, LVV06]. Expected benefits of the unit-norm constraint are to simplify the expression of the contrast function and to avoid continuums of minimizers that may complicate convergence analyses, reduce the convergence speed, or even destroy local convergence properties. The difficulty, though, is that constrained optimization is in general more challenging than unconstrained optimization. Nevertheless, the unit-norm constraint yields a feasible set that is merely a Cartesian product of (hyper)spheres. This feasible set has a natural structure of a Riemannian manifold, and the optimization problem can thus be tackled by Riemannian geometric techniques, exploited, e.g., in [SDH09,AG06,SH09,SK10]. The present chapter connects with the Riemannian approach in the sense that the proposed algorithm can be viewed as evolving on the Riemannian feasible set.

The second difficulty is that the range-based contrast function is *nonsmooth*, as it involves order statistics. Consequently, smooth Riemannian optimization techniques, as described in [AMS08], are not suitable for this task.

Optimization algorithms for minimizing range-based contrast functions were proposed in [Vri07] and in related papers, both in the orthogonal and nonorthogonal settings. In the orthogonal setting, orthonormality of the unmixing matrix $\mathbf{X}$ is enforced after prewhitening [HKO$^+$01]. This produces unmixed signals that are exactly uncorrelated. A method for orthogonal range-based ICA is proposed in [VLV07, §VI], where the independent components (ICs) are estimated sequentially (deflation approach) by a direct-search–type method in which the search frame is built by means of Jacobi-like rotations. A similar method, referred to as *support-width ICA* (SWICA), is proposed in [Vri07, §4.3].

The nonorthogonal setting, on the other hand, is motivated by the fact that the sources in real-world applications may not be exactly uncorrelated (and thus not exactly independent). It is even unreasonable to expect that the *finite-sample* estimation of the correlation between the

sources be exactly zero. While prewhitening remains a worthy preprocessing for the mixtures $\mathbf{M}$, relaxing the orthogonality constraint offers leeway to find an unmixing matrix $\mathbf{X}$ where the contrast function takes a lower value, with the hope that such an $\mathbf{X}$ will also yield a better recovery of the sources. This explains why nonorthogonal ICA methods have gained popularity in the recent years [DAK00, Yer02, Pha00, vdV01, BPR04].

In this direction, the nonorthogonal support-width ICA (NOSWICA) reported in [LVV06] relaxes the orthogonality constraint during the estimation process. To get around the difficulty of repeatedly estimating the same source in the deflation method, the first source is estimated by minimizing the support-width, and the rest of the sources are extracted by minimizing the penalized support.

Since there is an inherent drawback of accumulating the IC estimation error in the deflation method, an algorithm for simultaneous range-based ICA was proposed in [SCAA12], in the form of a population-based nonsmooth optimizer, namely, a cross-entropy method with von Mises-Fisher (vMF) distribution. The attractive feature here is the ability to intrinsically maintain the unit-norm constraint of randomly drawn candidate solutions from the vMF distribution.

The aforementioned optimization methods for range-based ICA are not supported by convergence analyses. In fact, as acknowledged in [Vri07, Remark 22], the first goal of Vrins's influential thesis was not to develop optimization schemes but rather to analyze the theoretical behavior of entropic contrast. Moreover, the cross-entropy method of [SCAA12] lacks convergence guarantees and suffers from the computational burden of such heuristics.

In this chapter, we contribute to improving the optimization aspect of range-based ICA algorithms by proposing and analyzing a new method for nonsmooth optimization with unit-norm constraint. The new method, when applied to the minimization of the range-based contrast function on the Cartesian product of spheres, results in a nonorthogonal ICA algorithm that is shown to strongly outperform other ICA algorithms in an image separation task.

The new optimization method, called SMADS, is an extension to the sphere of the mesh adaptive direct search (MADS). The MADS algorithm is an iterative derivative-free constrained optimizer proposed by Audet *et al.* [ACD07] to handle nonsmooth objective functions, where constraints are imposed using the extreme barrier approach—the objec-

tive function values of infeasible points are set to $+\infty$. It is an improvement over the generalized pattern search (GPS) class of methods [Tor97] that use only a finite number of *poll directions* for exploring the space of variables locally. The key advantage of the MADS algorithm over other frame-based methods, e.g., [CP00], is that it produces an asymptotically dense set of polling directions, meaning that convergence to a local optimum is guaranteed under mild assumptions. In other words, the MADS algorithm produces a Clarke stationary point. Aside from this merit, it is also easy to implement.

The proposed SMADS algorithm takes the form of a general-purpose derivative-free feasible method for unconstrained optimization on the sphere. By "feasible", we mean that all the iterates produced by SMADS belong to the sphere. By "unconstrained", we mean that, for simplicity of the exposition, the feasible set is assumed to be the whole sphere; however, since the workings of SMADS are only slightly more complex than those of MADS, it would not be difficult to handle further constraints akin to those considered in [ACD07]. (Observe that the original MADS is not suited for the sphere constraint, nor for manifold constraints in general. In particular, the hypertangent cone is empty everywhere and convergence results such as [ACD07, Th. 3.12] become vacuous.) The SMADS algorithm is built so as to retain all the convergence properties of the original MADS while remaining computationally inexpensive and easy to implement. The additional computational cost with respect to the original MADS is due to occasional scaling and rounding operations that occur during the face transition mechanism (described in Section 6.2); it remains comparatively negligible unless the objective function is extremely cheap to evaluate.

This chapter is organized as follows. Section 6.2 introduces notation that enables a concise description and analysis of the proposed algorithm. After recalling the fundamentals of the original MADS algorithm, Section 6.3 defines and discusses SMADS, the proposed mesh adaptive direct search algorithm on the sphere. An in-depth convergence analysis of SMADS is then proposed in Section 6.4, where the common principles and differences with the original MADS algorithm are highlighted. In Section 6.5, a practical instance of the SMADS algorithm is provided, extending the LTMADS algorithm to the sphere setting. The adaptation of the SMADS algorithm for the purpose of minimizing the range-based ICA contrast function is then detailed in Section 6.6. Numerical experiments are conducted in Section 6.7, where the task of unmixing the

image data is considered. Finally, conclusions and perspectives are given in Section 6.8.

6.2 Notation and Definitions

This section introduces notation, depicted in Figure 6.1, that enables a more concise description and analysis of the SMADS algorithm (Algorithm 6.1). This section can be skipped and referred to when needed.

Let $\mathbb{R}_0^n := \mathbb{R}^n \setminus \{\mathbf{0}\}$ denote the set of n-tuples of real numbers with the zero n-tuple excerpted, and let $\mathbb{R}_+$ denote the set of strictly positive real numbers. The unit (hyper)sphere in $\mathbb{R}^n$ is denoted by

$$\mathbb{S}^{n-1} := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}, \tag{6.1}$$

where $\|\cdot\|_2$ denotes the Euclidean norm, and the axis-aligned circumscribed (hyper)cube is denoted by

$$\mathcal{C}^{n-1} := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty = 1\}, \tag{6.2}$$

where $\|\cdot\|_\infty$ denotes the maximum norm. Let $s > 0$ with $\frac{1}{s}$ integer. The *virtual mesh* of size s on the hypercube $\mathcal{C}^{n-1}$ is the set

$$\mathcal{M}_s^{n-1} := \{\mathbf{x} \in \mathcal{C}^{n-1} : \frac{\mathbf{x}}{s} \in \mathbb{Z}^n\}. \tag{6.3}$$

We define the scaling map onto the sphere,

$$\mathrm{S} : \mathbb{R}_0^n \to \mathbb{S}^{n-1} : \mathbf{x} \mapsto \frac{\mathbf{x}}{\|\mathbf{x}\|_2}, \tag{6.4}$$

the scaling map onto the hypercube,

$$\mathrm{C} : \mathbb{R}_0^n \to \mathcal{C}^{n-1} : \mathbf{x} \mapsto \frac{\mathbf{x}}{\|\mathbf{x}\|_\infty}, \tag{6.5}$$

and the latter followed by rounding onto the virtual mesh of size s,

$$\mathrm{M}_s : \mathbb{R}_0^n \to \mathcal{M}_s^{n-1} : \mathbf{x} \mapsto s\lfloor \frac{1}{s}\,\mathrm{C}(\mathbf{x})\rceil, \tag{6.6}$$

where $\lfloor\cdot\rceil$ represents rounding off the argument to the nearest integer value.

Let $(\mathbf{e}_1, \ldots, \mathbf{e}_n)$ denote the canonical basis of $\mathbb{R}^n$. For $i = -1, \ldots, -n$, define $\mathbf{e}_i := -\mathbf{e}_{|i|}$. For $i = \pm 1, \ldots, \pm n$ and $\mathbf{x} \in \mathbb{R}^n$, define $x_i := \mathbf{e}_i^\top \mathbf{x}$. For $i = \pm 1, \ldots, \pm n$, let $F_{\mathbf{e}_i}$ denote the face of $\mathcal{C}^{n-1}$ that contains $\mathbf{e}_i$, that is,

$$F_{\mathbf{e}_i} := \{\mathbf{x} \in \mathcal{C}^{n-1} : x_i = 1\}.$$

The affine hull of a face $F_{\mathbf{e}_i}$ is $\operatorname{aff}(F_{\mathbf{e}_i}) := \{\mathbf{x} \in \mathbb{R}^n : \boldsymbol{e}_i^\top \boldsymbol{x} = 1\} = \{\mathbf{x} \in \mathbb{R}^n : x_i = 1\}$. This is the hyperplane that contains $F_{\mathbf{e}_i}$. The tangent space of $\operatorname{aff}(F_{\boldsymbol{e}_i})$ is $\operatorname{T}\operatorname{aff}(F_{\boldsymbol{e}_i}) = \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{e}_i^\top \boldsymbol{v} = 0\}$. It is the hyperplane spanned by $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{i-1}, \boldsymbol{e}_{i+1}, \ldots, \boldsymbol{e}_n\}$. For $i = \pm 1, \ldots, \pm n$ and $\mathbf{x} \in \mathbb{R}^n$ with $x_i \neq 0$, define the scaling of $\mathbf{x}$ onto $\operatorname{aff}(F_{\mathbf{e}_i})$ by

$$(\mathbf{x})_{\uparrow i} := \frac{\mathbf{x}}{x_i}.$$

Observe that $(\mathbf{x})_{\uparrow i}$ is the intersection between $\operatorname{aff}(F_{\mathbf{e}_i})$ and the straight line through $\mathbf{0}$ and $\mathbf{x}$. Note also that for any point $\boldsymbol{x}$ on the affine hull $\operatorname{aff}(F_{\mathbf{e}_i})$ of a given face $F_{\mathbf{e}_i}$, the scaling $\mathrm{C}(\boldsymbol{x})$ leaves the point $\boldsymbol{x}$ unmodified if $\boldsymbol{x} \in F_{\mathbf{e}_i}$ and brings $\boldsymbol{x}$ to the nearest face $F_{\mathbf{e}_j}$ otherwise. The scaling C—and hence also the scaling M_s—thus implicitly offers a convenient face transition mechanism that will be used to produce iterates belonging to $\mathcal{C}^{n-1}$ at all times.

For $i = \pm 1, \ldots, \pm n$ and $\mathbf{d} \in \mathbb{R}^{n-1}$, define the embedding $(\mathbf{d})_{\hookrightarrow i}$ to be the vector of $\mathbb{R}^n$ obtained from $\mathbf{d}$ by inserting a zero in the ith position. In other words, $(\mathbf{d})_{\hookrightarrow i} \in \mathbb{R}^n$ with

$$\mathbf{e}_j^\top (\mathbf{d})_{\hookrightarrow i} = \begin{cases} d_j & \text{for } j = 1, \ldots, i-1 \\ 0 & \text{for } j = i \\ d_{j-1} & \text{for } j = i+1, \ldots, n. \end{cases}$$

Given a sequence $\{\alpha_k\}_{k \geq 0} \in \mathbb{R}$, we let the smallest α encountered up to element k be denoted by

$$\underline{\alpha}_k := \min_{k' \leq k} \alpha_{k'}. \tag{6.7}$$

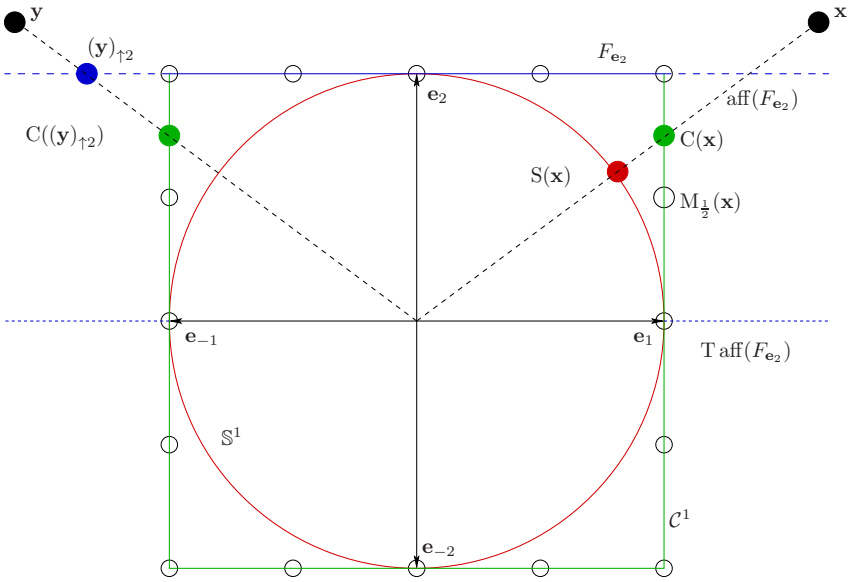


Figure 6.1: A pictorial representation of the introduced notation for a 2-dimensional setting, which illustrates the unit sphere $\mathbb{S}^1$, the hypercube $\mathcal{C}^1$ (alternately long- and short-dashed lines), the virtual mesh $\mathcal{M}^1_{\frac{1}{2}}$ of size $\frac{1}{2}$ (small hollow dots), one of the four faces ($F_{\mathbf{e}_2}$ in alternately long- and triple short-dashed line), its affine hull $\mathrm{aff}(F_{\mathbf{e}_2})$ (medium-dashed lines) and the corresponding tangent space $\mathrm{T}\,\mathrm{aff}(F_{\mathbf{e}_2})$ (short-dashed lines). A point $\mathbf{x} \in \mathbb{R}^2_0$ is depicted alongside its scaling $\mathrm{S}(\boldsymbol{x})$ onto the sphere (checked dot) and its scaling $\mathrm{C}(\boldsymbol{x})$ onto the hypercube (dull gray dot on the right). The scaling-and-rounding map $\mathrm{M}_{\frac{1}{2}}$ of $\boldsymbol{x}$ to the virtual mesh is also shown (hollow dot). Another point $\mathbf{y} \in \mathbb{R}^2_0$ is depicted together with its scaling $(\boldsymbol{y})_{\uparrow 2}$ (dark gray dot) onto the affine hull of the face $F_{\mathbf{e}_2}$. A face switching mechanism is implicitly present in the mapping C as typified by the point $\mathrm{C}((\mathbf{y})_{\uparrow 2})$ (dull gray dot on the left).

6.3 SMADS Algorithm on the Sphere

The purpose of this section is to define and discuss SMADS, the proposed mesh adaptive direct search algorithm on the sphere.

We first recall the general ideas behind the original MADS algorithm. Initially presented in [ACD07], MADS is an iterative derivative-free constrained optimization algorithm generating a finite number of trial points at each iteration k, using a set of directions around the current iterate $\boldsymbol{x}_k$. The current objective value $f(\boldsymbol{x}_k)$ is compared to the objective function at the feasible trial points and is updated when improvement is achieved (without any sufficient decrease requirement). The trial points must belong to a specific *mesh*, constructed using a set of directions in $\mathbb{R}^n$ scaled by the *mesh size* parameter. Note that in order to ensure the MADS convergence properties, these directions must respect some conditions that are detailed in [ACD07], and will be discussed later in the convergence analysis of SMADS (Section 6.4).

Each iteration of the MADS algorithm is composed of two steps, namely, the *search step* and the *poll step*. The search step is optional and allows the exploration of the search space. Any heuristic can be used to sample candidates, as long as they lie on the mesh. This step is not involved in the convergence analysis and is typically used to enhance global exploration of the search space. The poll step on the other hand is mandatory, and is used whenever the search step is not used or fails to produce an improved iterate. The poll step is the one that leads to the convergence results of the MADS algorithm. To do so, it only produces candidate points that lie on a subset of the mesh, called the *poll frame*, to which corresponds a parameter called the *poll size*. The mesh size parameter is then updated at the end of each iteration and the poll size is updated accordingly.

We now present the SMADS algorithm, our adaptation of MADS to the sphere setting. The algorithm statement is given in Algorithm 6.1. Several comments and precisions follow.

The problem addressed by SMADS is to minimize a given real-valued function f defined on the (hyper)sphere $\mathbb{S}^{n-1}$. We use the same notation f to denote its zero-degree homogeneous extension to $\mathbb{R}_0^n$, i.e.,

$$f(\mathbf{x}) := f(\mathrm{S}(\mathbf{x})), \tag{6.8}$$

where S denotes the scaling to the sphere given by (6.4).

Algorithm 6.1 : Spherical MADS (SMADS)

Refer to Section 6.2 for notation
Input: real-valued function f on the sphere $\mathbb{S}^{n-1}$, extended to $\mathbb{R}_0^n$ according to (6.8); convergence threshold $\varepsilon \geq 0$ (where $\varepsilon = 0$ generates an infinite sequence); integer $\tau > 1$;
Initialization: $k := 0$; initial iterate $\mathbf{x}_0 := \pm\mathbf{e}_j$ for some $j \in \{1, \ldots, n\}$; mesh size parameter $\alpha_0^m := \tau^{\beta_0}$ for some integer $\beta_0 \leq 0$; poll size parameter α_0^p as in (6.11);
while $\alpha_k^p > \varepsilon$ **do**
SEARCH STEP (*optional*)
    Generate $N \geq 0$ points $\mathbf{x}_k^1, \ldots, \mathbf{x}_k^N$ on the mesh $\mathcal{M}_{\underline{\alpha}_k^m}^{n-1}$ (see (6.3) and (6.7));
    if $N > 0$ **and** there is $i^* \in \{1, \ldots, N\}$ such that $f(\mathbf{x}_k^{i^*}) < f(\mathbf{x}_k)$ **then**
        $\mathbf{x}_{k+1} := \mathbf{x}_k^{i^*}$ and declare iteration successful;
    else
P
        Construct poll frame P_k according to Definition 6.1; (A specific way of constructing P_k will be given in Section 6.5.1.)
        if there is $\mathbf{p}^* \in P_k$ such that $f(\mathbf{p}^*) < f(\mathbf{x}_k)$ **then**
            $\mathbf{x}_{k+1} := \mathbf{p}^*$ and declare iteration successful;
        else
            $\mathbf{x}_{k+1} := \mathbf{x}_k$ and declare iteration unsuccessful;
        end if
    end if
    $k := k + 1$;
    Update the parameters α_k^m as in (6.10) and α_k^p as in (6.11);
end while
return $\mathring{\mathbf{x}} := \mathrm{S}(\mathbf{x}_k) := \mathbf{x}_k / \|\mathbf{x}_k\|_2$;

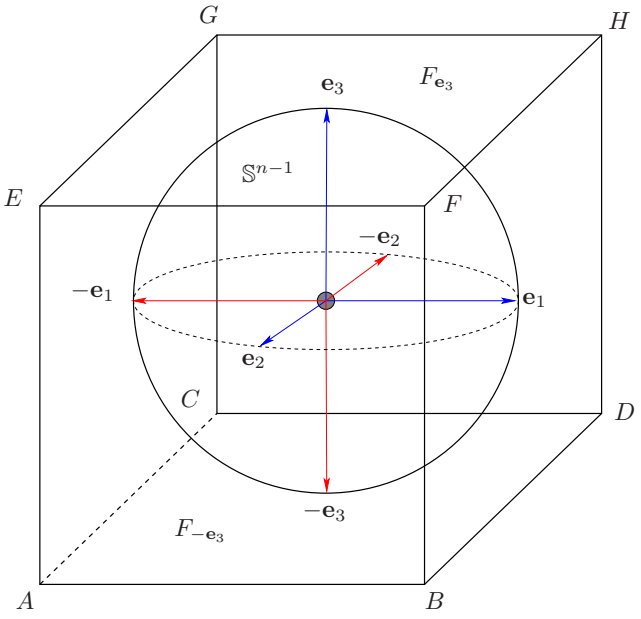


Figure 6.2: A pictorial representation of the algorithm setting, where the unit sphere $\mathbb{S}^{n-1}$ is enclosed within a hypercube $\mathcal{C}^{n-1}$. All the iterates lie on the (bounded) faces $F_{\mathbf{e}_i}$, $i = \pm 1, \ldots, \pm n$, of $\mathcal{C}^{n-1}$, e.g., the faces labeled as $ABCD$ and $EFGH$ are $F_{-\mathbf{e}_3}$ and $F_{\mathbf{e}_3}$, respectively; $\mathbf{e}_i$ are the endpoints of the canonical basis vectors (dashed and continuous arrows) pointing outward from the origin of $\mathbb{S}^{n-1}$.

All the iterates generated by SMADS belong to the circumscribed hypercube $\mathcal{C}^n$. SMADS can however be viewed as an iteration on the sphere $\mathbb{S}^{n-1}$ because the scaling map S restricted to $\mathcal{C}^n$ is a one-to-one map that leaves f invariant. The situation is depicted in Figure 6.2. By working on the hypercube instead of the sphere, we benefit from the fact that the search space is locally a linear manifold almost everywhere, but we need a face-transition mechanism (provided by the scaling map M_s (6.6)). As we will show below, the face-transition mechanism present in SMADS introduces a very mild computational overhead and retains all the convergence properties of the original MADS.

The search step in Algorithm 6.1 is optional since $N = 0$ is allowed. In view of (6.3) and (6.7), the finest virtual mesh (6.3) at any iteration

k is given by $\mathcal{M}_{\underline{\alpha}_k^m}^{n-1} = \mathcal{C}^{n-1} \cap \underline{\alpha}_k^m \mathbb{Z}^n$. As we will see, $1/\underline{\alpha}_k^m$ is an integer at every iteration k, and it follows that $\mathcal{M}_{\underline{\alpha}_k^m}^{n-1}$ consists of the nodes of a fitted grid on the circumscribed hypercube $\mathcal{C}^{n-1}$. This is illustrated in Figure 6.1 (for $n = 2$) and Figure 6.3 (for $n = 3$) for one face.

The poll step is performed whenever the search step does not produce a new iterate. Since the mesh $\mathcal{M}_{\underline{\alpha}_k^m}^{n-1}$ is finite and the search step does not update α_k^m, it follows that the poll step is performed infinitely often (when the convergence threshold ε is set to 0).

The poll frame P_k is constructed around the current iterate $\boldsymbol{x}_k$, according to Definition 6.1.

Definition 6.1 (poll frame). Given a positive spanning set $\boldsymbol{\Xi}_k \subset \mathbb{R}^{n-1}$ (i.e., every element of $\mathbb{R}^{n-1}$ can be written as a nonnegative linear combination of the elements of $\boldsymbol{\Xi}_k$), the *poll frame* of SMADS at the current iterate $\mathbf{x}_k \in F_{\mathbf{e}_i}$ is the set

$$P_k := \mathrm{M}_{\underline{\alpha}_k^m} \left(\{ \mathbf{x}_k + \alpha_k^m \, (\boldsymbol{\xi})_{\hookrightarrow i(k)} : \boldsymbol{\xi} \in \boldsymbol{\Xi}_k \} \right), \tag{6.9}$$

where M (6.6) is the mapping to the virtual mesh and $i(k)$ is such that $\mathbf{x}_k \in F_{\mathbf{e}_{i(k)}}$. (When $\mathbf{x}_k$ belongs to more than one face, the ambiguity can be removed by setting $i(k)$ to be the smallest i such that $\mathbf{x}_k \in F_{\mathbf{e}_i}$.) Furthermore, each *polling direction* $\boldsymbol{\xi} \in \boldsymbol{\Xi}_k$ must satisfy the conditions below:

- $\boldsymbol{\xi} \in \mathbb{Z}_0^{n-1}$;
- $\alpha_k^m \, \|\boldsymbol{\xi}\| \leq \alpha_k^p$.

Observe also that $\| \, (\boldsymbol{\xi})_{\hookrightarrow i(k)} \, \| = \|\boldsymbol{\xi}\|$. In the original MADS algorithm presented in [ACD07], there is more freedom on the nature of the vectors $\boldsymbol{\xi}$; for simplicity, we adopt the choice made in the LTMADS method, see [ACD07, §4].

(The reader familiar with the MADS algorithm will notice that the last bullet of [ACD07, Def. 2.2] has no counterpart in Definition 6.1. Translated in the SMADS context, the condition requires that limits of the normalized sets $\{ \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} : \boldsymbol{\xi} \in \boldsymbol{\Xi}_k \}$ are positive spanning sets. This condition is useful when f is smooth and allows to conclude, as in [CP00, Th. 4.2], that each cluster point is a stationary point of f. However, as discussed in [ACD07], when f is nonsmooth, other assumptions are needed to conclude that cluster points are (Clarke) stationary points.

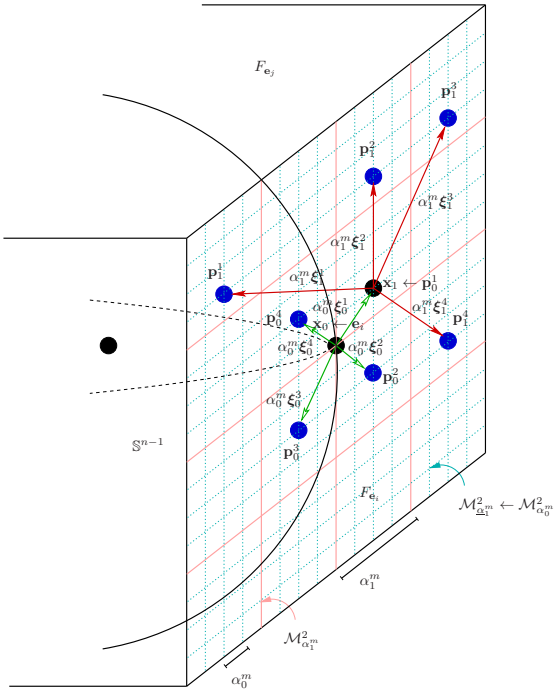


Figure 6.3: The initial iterate $\mathbf{x}_0$ has been selected as $\mathbf{e}_i$. As per (6.3), the mesh is conceptually built around $\mathbf{x}_0$, which is an integer lattice (dashed grid) with the mesh size parameter α_0^m. For the poll directions $\boldsymbol{\Xi}_0 = \{\boldsymbol{\xi}_0^1, \ldots, \boldsymbol{\xi}_0^4\}$ (green arrows with a hollow head for $k = 0$), the algorithm generates trial points $\mathbf{p}_0^1, \ldots, \mathbf{p}_0^4$ (dark blue dots) around $\mathbf{x}_0$, which constitute the poll frame denoted by a set P_0. Note that at $k = 0$, the mesh points located at the intersections of the dashed horizontal and vertical lines form a set $\mathcal{M}^2_{\underline{\alpha}_0^m}$, such that $P_0 \subset \mathcal{M}^2_{\underline{\alpha}_0^m}$ as specified in Definition 6.1. Since $f(\mathbf{p}_0^1) < f(\mathbf{x}_0)$, $\mathbf{p}_0^1$ is used to update the current iterate $\mathbf{x}_0$ as $\mathbf{x}_1$. Recall that the mesh $\mathcal{M}^2_{\underline{\alpha}_1^m}$ with the mesh size parameter $\underline{\alpha}_1^m$ given by (6.7) is the same as $\mathcal{M}^2_{\alpha_0^m}$, owing to the fact that $\alpha_1^m = \tau^1 \alpha_0^m$ where $\tau > 1$ is a fixed integer. At every iteration k, four trial points are generated as the set of directions $\boldsymbol{\Xi}_k$ (arrows with a solid head for $k = 1$) considered here is the maximal positive basis with $\#\boldsymbol{\Xi}_k = 2(n-1) = 4$.

The denseness assumption that we will encounter in Theorem 6.4 makes the positive spanning limit set condition superfluous.)

The mesh size parameter α_k^m is updated at the end of each iteration according to the following rule: given two integers $\beta^- \leq -1$ and $\beta^+ \geq 0$, choose

$$\alpha_k^m = \tau^{\beta_k} \alpha_{k-1}^m \tag{6.10}$$

for some

$$\beta_k = \begin{cases} \{0, 1, \ldots, \beta^+\} & \text{if iteration is successful} \\ \{\beta^-, \beta^- + 1, \ldots, -1\} & \text{otherwise,} \end{cases}$$

such that $\alpha_{k+1}^m \leq 1$. (Recall that $\tau > 1$ is a fixed integer.) Hence, for all k, $\alpha_k^m = \tau^{\sum_{i=0}^k \beta_i}$ and $\underline{\alpha}_k^m = \tau^{\min_{0 \leq k' \leq k} \sum_{i=0}^{k'} \beta_i}$, where $\min_{0 \leq k' \leq k} \sum_{i=0}^{k'} \beta_i \leq \sum_{i=0}^k \beta_i \leq 0$ are integers. In particular, as announced, $1/\underline{\alpha}_k^m$ is an integer for all k.

The poll size parameter α_k^p is then determined such that $\alpha_k^m \leq \alpha_k^p \leq 1$ and moreover, for every infinite index set K, $\lim_{k \in K} \alpha_k^m = 0$ if and only if $\lim_{k \in K} \alpha_k^p = 0$. To fix ideas, we set

$$\alpha_k^p = \sqrt{\alpha_k^m} \tag{6.11}$$

which satisfies both conditions. (We impose that $\alpha_k^p \leq 1$ in order to be able to apply the last point of Lemma 6.1 in Theorem 6.5.)

As we will see in Theorem 6.1, $\lim_{k \to \infty} \alpha_k^p = 0$, which justifies the stopping criterion. A crucial aspect of SMADS is that all the iterates and all the poll points belong to the virtual mesh defined in (6.3), as shown next.

Proposition 6.1. In SMADS (Algorithm 6.1), for all k, and recalling definitions (6.3) and (6.7), it holds that

$$\mathbf{x}_k \in \mathcal{M}_{\underline{\alpha}_k^m}^{n-1} \quad \text{and} \quad P_k \subset \mathcal{M}_{\underline{\alpha}_k^m}^{n-1}.$$

Proof. The claim is direct for P_k in view of the presence of the scaling-and-rounding map $\mathrm{M}_{\underline{\alpha}_k^m}$ in (6.9). Regarding $\mathbf{x}_k$, let k' be the smallest integer such that $\mathbf{x}_{k'} = \mathbf{x}_k$. If $k' = 0$, then $\mathbf{x}_k$ is a face center (i.e. $\boldsymbol{x}_k = \boldsymbol{e}_i, -n \leq i \leq n$) and the claim follows, so we henceforth assume that $k' > 0$. Then $\mathbf{x}_{k'}$ is a new point generated by Algorithm 6.1 at step $k' - 1$ and, in view of the workings of the algorithm, we have that $\mathbf{x}_{k'} \in \mathcal{M}_{\underline{\alpha}_{k'-1}^m}^{n-1}$. In view of the mesh size update rule (6.10), we have that

$\underline{\alpha}^m_{k'-1}/\underline{\alpha}^m_k$ is an integer and hence that $\mathcal{M}^{n-1}_{\underline{\alpha}^m_{k'-1}} \subseteq \mathcal{M}^{n-1}_{\underline{\alpha}^m_k}$. The claim follows. □

An important consequence is that $\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i(k)}$ involved in the definition (6.9) of P_k is such that $\frac{1}{\underline{\alpha}^m_k}\left(\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i(k)}\right)$ is a vector of integers for all k. Hence, in the frequently encountered situation where $\alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i(k)}$ is sufficiently small for $\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i(k)}$ to remain on the hypercube $\mathcal{C}^{n-1}$, we have that the scaling-and-rounding operation $\mathrm{M}_{\underline{\alpha}^m_k}$ in P_k (6.9) has no effect and can thus be omitted. When it cannot be omitted, the scaling-and-rounding operation remains computationally cheap. This justifies the claim made in the introduction that the computational overhead of SMADS with respect to MADS is minor.

6.4 Convergence Analysis of SMADS

In adapting to SMADS the convergence analysis of MADS [ACD07, §3], the main difficulty is to take face transition into account. In other words, we need to address the fact that, whenever $\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i(k)}$ in the definition (6.9) of P_k is outside the hypercube $\mathcal{C}^{n-1}$, the action of $\mathrm{M}_{\underline{\alpha}^m_k}$ scales it back to $\mathcal{C}^{n-1}$ and rounds it to the nearest point of the virtual mesh $\mathcal{M}^{n-1}_{\underline{\alpha}^m_k}$.

We first show that the convergence analysis of MADS still holds if the notion of direction is replaced by the notion of *adapted direction*, defined next. We conclude by showing that if the set of refining directions is dense, then so is the set of adapted refining directions, under a reasonable condition on the size of the poll frame.

The scaling-and-rounding operation that occurs during face transition prompts us to define *adapted* search directions as follows. The concept is illustrated in Figure 6.4.

Definition 6.2 (adapted search direction). Let $\mathbf{x}_k \in F_{\mathbf{e}_i}$ be the current iterate and let $\boldsymbol{\xi} \in \mathbb{R}^{n-1}$ be a search direction, where i is chosen as in Definition 6.1 if there is an ambiguity. The *adapted search direction* $\hat{\boldsymbol{\xi}}$ is the direction of the variation of $\mathbf{x}_k$ obtained by going to $\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i}$, scaling to the hypercube, rounding onto the virtual mesh, and scaling back to the affine hull of face $F_{\mathbf{e}_i}$; more precisely,

$$\hat{\boldsymbol{\xi}} := \frac{1}{\alpha^m_k}\left(\left(\mathrm{M}_{\underline{\alpha}^m_k}(\mathbf{x}_k + \alpha^m_k (\boldsymbol{\xi})_{\hookrightarrow i})\right)_{\uparrow i} - \mathbf{x}_k\right). \tag{6.12}$$

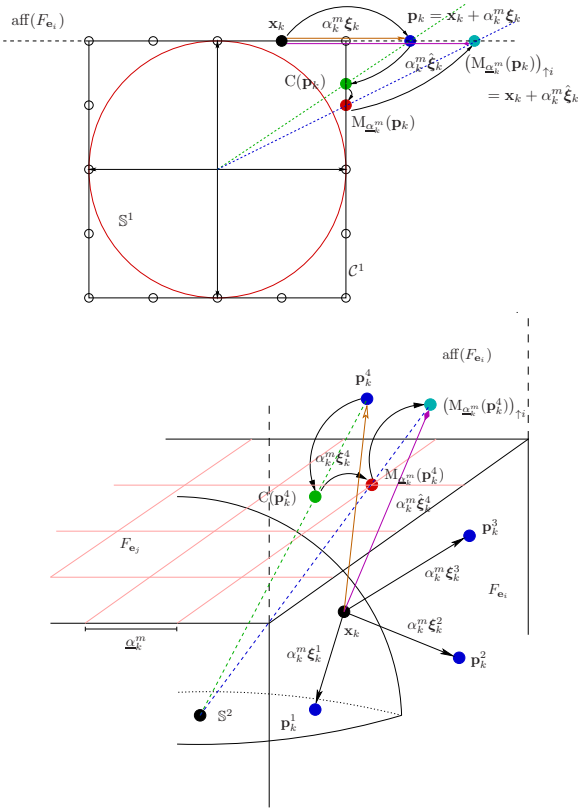


Figure 6.4: Illustration of the face switching mechanism and the notion of adapted direction (top: 2D; bottom: 3D). At iteration k, a poll direction (arrow with a hollow head—top: $\boldsymbol{\xi}_k$; bottom: $\boldsymbol{\xi}_k^4$) produces the frame point (dark gray dot—top: $\mathbf{p}_k$; bottom: $\mathbf{p}_k^4$) that belongs to aff($F_{\mathbf{e}_i}$) but not to $F_{\mathbf{e}_i}$. Consequently, it must be assigned to a mesh point on the face $F_{\mathbf{e}_j}$ of $\mathcal{C}^{n-1}$. This is carried out in two steps: first, by scaling onto the hypercube, symbolized as C, the frame point is restricted to $\mathcal{C}^{n-1}$ (dull gray dot); next, it is assigned to the nearest mesh point using the operation $\mathrm{M}_{\underline{\alpha}_k^m}$ (hollow dot). By scaling this point back to aff($F_{\mathbf{e}_i}$) denoted as $(\cdot)_{\uparrow i}$ (checked dot), the face switching procedure results in an adapted search direction (arrow with a diamond head—top: $\hat{\boldsymbol{\xi}}_k$; bottom: $\hat{\boldsymbol{\xi}}_k^4$).

Observe that if $\boldsymbol{\xi}$ is an integer vector and $\mathbf{x}_k + \alpha_k^m (\boldsymbol{\xi})_{\hookrightarrow i}$ remains in $F_{\mathbf{e}_i}$, then $\hat{\boldsymbol{\xi}} = \boldsymbol{\xi}$. Observe also that

$$P_k = \mathrm{C}\left(\{\mathbf{x}_k + \alpha_k^m \hat{\boldsymbol{\xi}} : \boldsymbol{\xi} \in \boldsymbol{\Xi}_k\}\right);$$

hence $\{\mathbf{x}_k + \alpha_k^m \hat{\boldsymbol{\xi}} : \boldsymbol{\xi} \in \boldsymbol{\Xi}_k\}$ is the frame in $\mathrm{aff}(F_{\mathbf{e}_i})$ that projects to P_k by scaling onto $\mathcal{C}^{n-1}$.

Until Section 6.4.1, the analysis of SMADS unfolds much as the analysis of MADS in [ACD07, §3], with directions replaced by adapted directions. We provide some details for the reader's convenience.

THEOREM 6.1. Under the assumption that a locally Lipschitz continuous function $f : \mathbb{S}^{n-1} \to \mathbb{R}$ and an initial iterate $\mathbf{x}_0 \in F_{\mathbf{e}_i}$ are available for the algorithm,

(i) all the iterates $\{\mathbf{x}_k\}$ produced by the SMADS algorithm lie in a compact set;

(ii) the update of the poll and mesh size parameters satisfy

$$\liminf_{k\to\infty} \alpha_k^p = \liminf_{k\to\infty} \alpha_k^m = 0.$$

Proof. The first claim is straightforward: all the iterates belong to the hypercube $\mathcal{C}^{n-1}$, which is a compact set.

The second claim can be shown by means of a contradiction argument. We know that for all $N \geq 0$, the iterates $\mathbf{x}_0, \ldots, \mathbf{x}_N$ belong to the mesh $\mathcal{M}^{n-1}_{\underline{\alpha}_N^m}$. Suppose, by way of contradiction, that the second claim does not hold. Then $\underline{\alpha}_\infty^m := \lim_{N\to\infty} \underline{\alpha}_N^m > 0$, and it follows that all the iterates belong to $\mathcal{M}^{n-1}_{\underline{\alpha}_\infty^m}$. This set contains finitely many points, hence an improved mesh point can be found only finitely many times. After that, the mesh size parameter $\underline{\alpha}_N^m$ is decreased at each iterate according to rule (6.10), which implies that the mesh size parameter goes to zero, a contradiction. □

DEFINITION 6.3 (minimal frame center). If the poll step fails to produce an improved mesh point at iteration k, then the iterate $\mathbf{x}_k$ is said to be a *minimal frame center*.

DEFINITION 6.4 (refining subsequence). Let $\{\mathbf{x}_k\}$, $\{\alpha_k^p\}$ and $\{i(k)\}$ be generated by Algorithm 6.1. If K is an infinite index set such that $\{\mathbf{x}_k\}_{k\in K}$ are minimal frame centers, $\{\alpha_k^p\}_{k\in K}$ converges to zero, and $\{i(k)\}_{k\in K}$ is a constant, say j, then $\{\mathbf{x}_k\}_{k\in K}$ is called a *refining subsequence* (on face j).

The next result concerning the existence of at least one convergent refining subsequence can be viewed as an extension of Theorem 3.6 in [AD02] obtained for the GPS algorithms.

THEOREM 6.2. Let $\{\mathbf{x}_k\}$ be the iterates produced by the SMADS algorithm optimizing a locally Lipschitz $f : \mathbb{S}^{n-1} \to \mathbb{R}$, then there exists at least one convergent refining subsequence.

Proof. The result follows immediately from Theorem 6.1 that guarantees the existence of a subset of indices $K' \subset K$ for which $\{\alpha_k^m\}_{k\in K'} \downarrow 0$ with K being the set of indices corresponding to the minimal frame centers. □

In SMADS, the concept of refining direction [ACD07, Def. 3.2] is complemented by the one of *adapted* refining direction.

DEFINITION 6.5 (refining direction and adapted refining direction). Let $\mathring{\mathbf{x}}$ be the limit of a convergent refining subsequence. For some subset of indices $L \subseteq K$, if the limit $\lim_{k\in L} \frac{\boldsymbol{\xi}_k}{\|\boldsymbol{\xi}_k\|}$ exists with poll direction $\boldsymbol{\xi}_k \in \boldsymbol{\Xi}_k$, then this limit is said to be a *refining direction* for $\mathring{\mathbf{x}}$. For some subset of indices $L \subseteq K$, if the limit $\lim_{k\in L} \frac{\hat{\boldsymbol{\xi}}_k}{\|\hat{\boldsymbol{\xi}}_k\|}$ exists with poll direction $\boldsymbol{\xi}_k \in \boldsymbol{\Xi}_k$ (where the hat denotes the adapted direction as in (6.12)), then this limit is said to be an *adapted refining direction* for $\mathring{\mathbf{x}}$.

The subsequent theorem is the generalization of [ACD07, Th. 3.12] by taking into account adapted refining directions. We first recall a classical definition.

DEFINITION 6.6 (Clarke generalized directional derivative). The *Clarke generalized directional derivative* of f at $\mathbf{x} \in \mathbb{R}_0^n$ along a direction $\mathbf{v} \in \mathbb{R}^n$ is defined as

$$f^\circ(\mathbf{x};\mathbf{v}) := \limsup_{\mathbf{y}\to\mathbf{x},\ t\downarrow 0} \frac{f(\mathbf{y}+t\mathbf{v}) - f(\mathbf{y})}{t}.$$

THEOREM 6.3. Let f (as defined in (6.8)) be Lipschitz near a limit $\mathring{\mathbf{x}} \in F_{\mathbf{e}_j}$ of a refining subsequence on face j, and $\hat{\mathbf{v}} \in \mathrm{T}\,\mathrm{aff}(F_{\mathbf{e}_j})$ be an adapted refining direction for $\mathring{\mathbf{x}}$. Then the generalized directional derivative of f at $\mathring{\mathbf{x}}$ in the direction $\hat{\mathbf{v}}$ is nonnegative, i.e., $f^\circ(\mathring{\mathbf{x}};\hat{\mathbf{v}}) \geq 0$.

Proof. The proof reproduces the proof of [ACD07, Th. 3.12], with $\mathbb{R}^n$ replaced by the linear manifold $\mathrm{aff}(F_{\mathbf{e}_j})$ and without constraints.

Let $\{\mathbf{x}_k\}_{k\in K}$ be a refining subsequence that converges to $\mathring{\mathbf{x}}$. Express $\hat{\mathbf{v}} \in \mathrm{T}\,\mathrm{aff}(F_{\mathbf{e}_j})$ as $\lim_{k\in L} \frac{\hat{\boldsymbol{\xi}}_k}{\|\hat{\boldsymbol{\xi}}_k\|}$ with $\boldsymbol{\xi}_k \in \boldsymbol{\Xi}_k$ for all $k \in L$, following Definition 6.5. In view of [ACD07, Prop. 3.9] (which obviously generalizes from $\mathbb{R}^n$ to linear manifolds), we have that

$$f^\circ(\mathring{\mathbf{x}}; \hat{\mathbf{v}}) = \lim_{k\in L} f^\circ \left(\mathring{\mathbf{x}}; \frac{\hat{\boldsymbol{\xi}}_k}{\left\|\hat{\boldsymbol{\xi}}_k\right\|} \right). \tag{6.13}$$

The SMADS algorithm guarantees that $\alpha_k^m \left\|\hat{\boldsymbol{\xi}}_k\right\|$ converges to zero due to the following rationale:

(i) $\alpha_k^m \|\boldsymbol{\xi}\|_\infty \leq \alpha_k^p$ (second bullet in the conditions to be satisfied by the poll frame in Definition 6.1);

(ii) $\liminf_{k\to\infty} \alpha_k^p = 0$ (second bullet in Theorem 6.1).

By the definition of Clarke's generalized directional derivative, we have

$$\begin{aligned} f^\circ(\mathring{\mathbf{x}}; \hat{\mathbf{v}}) &\geq \limsup_{k\in L} \frac{f(\mathbf{x}_k + \alpha_k^m \left\|\hat{\boldsymbol{\xi}}_k\right\| \frac{\hat{\boldsymbol{\xi}}_k}{\|\hat{\boldsymbol{\xi}}_k\|}) - f(\mathbf{x}_k)}{\alpha_k^m \left\|\hat{\boldsymbol{\xi}}_k\right\|} \\ &= \limsup_{k\in L} \frac{f(\mathbf{x}_k + \alpha_k^m \hat{\boldsymbol{\xi}}_k) - f(\mathbf{x}_k)}{\alpha_k^m \left\|\hat{\boldsymbol{\xi}}_k\right\|} \\ &\geq 0. \end{aligned} \tag{6.14}$$

The first inequality stems from the fact that $f^\circ(\mathring{\mathbf{x}}; \hat{\mathbf{v}})$ given in Definition 6.6 uses a more general supremum limit, whereas the one on the right hand side of (6.14) considers a particular choice. The last inequality holds because $\mathbf{x}_k$ is a minimal frame center. □

Much as for MADS, we have the following convergence result to Clarke stationary points.

Definition 6.7 (Clarke stationary point on the sphere). Let f be Lipschitz continuous near $\mathring{\mathbf{x}} \in F_{\mathbf{e}_j}$. Then $\mathring{\mathbf{x}}$ is termed a *Clarke stationary point* of f if $f^\circ(\mathring{\mathbf{x}}; \mathbf{v}) \geq 0$ for every direction $\mathbf{v} \in \mathbb{R}^n$. Since f is homogeneous of degree zero, this is equivalent to $f^\circ(\mathring{\mathbf{x}}; \mathbf{v}) \geq 0$ for every direction $\mathbf{v} \in \mathrm{T}\,\mathrm{aff}(F_{\mathbf{e}_j})$.

THEOREM 6.4. Let f (as defined in (6.8)) be Lipschitz near a limit $\mathring{\mathbf{x}} \in F_{\mathbf{e}_j}$ of a refining subsequence on face j. If the set of adapted refining directions for $\mathring{\mathbf{x}}$ is dense in $F_{\mathbf{e}_j}$, then $\mathring{\mathbf{x}}$ is a Clarke stationary point of f.

Proof. The proof follows from Theorem 6.3 and [ACD07, Prop. 3.9]. □

6.4.1 Denseness of Adapted Refining Directions

We now set out to show that, under some condition, if the set of refining directions is dense, then so is the set of adapted refining directions.

To this end, the next lemma will play a crucial role. It provides bounds on the perturbation incurred due to the scaling-and-rounding operation of the mapping M_s.

LEMMA 6.1. Let $\mathbf{y}$ belong to $\mathrm{aff}(F_{\mathbf{e}_i})$ for some $i \in \{\pm 1, \ldots, \pm n\}$; in other words, $\mathbf{e}_i^\top \mathbf{y} = 1$. Then, for all mesh sizes $s > 0$ with $\frac{1}{s}$ integer, we have

$$\| \mathrm{M}_s(\mathbf{y}) - \mathrm{C}(\mathbf{y}) \|_\infty \leq \frac{s}{2}. \tag{6.15}$$

and

$$\| \left(\mathrm{M}_s(\mathbf{y})\right)_{\uparrow i} - \mathbf{y} \|_\infty \leq s \frac{1}{|(\mathrm{M}_s(\mathbf{y}))_i (\mathrm{C}(\mathbf{y}))_i|}. \tag{6.16}$$

If moreover $s < \frac{1}{\|\mathbf{y}\|_\infty}$, then

$$\| \left(\mathrm{M}_s(\mathbf{y})\right)_{\uparrow i} - \mathbf{y} \|_\infty \leq 2s \|\mathbf{y}\|_\infty^2. \tag{6.17}$$

Finally, if moreover $\|\mathbf{y}\|_\infty \leq 2$, then

$$\| \left(\mathrm{M}_s(\mathbf{y})\right)_{\uparrow i} - \mathbf{y} \|_\infty \leq 8s. \tag{6.18}$$

Proof. Claim (6.15) is straightforward from the definitions and from basic rounding error theory.

We show (6.16). To simplify the notation, let $\mathbf{a} := \mathrm{M}_s(\mathbf{y})$ and $\mathbf{b} := \mathrm{C}(\mathbf{y})$, and observe that $(\mathrm{M}_s(\mathbf{y}))_{\uparrow i} - \mathbf{y} = (\mathbf{a})_{\uparrow i} - (\mathbf{b})_{\uparrow i}$. Observe also that $|a_i| \leq \|\mathbf{a}\|_\infty \leq 1$ and $|b_i| \leq \|\mathbf{b}\|_\infty \leq 1$; for $\mathbf{b}$, this is because $\mathbf{b} = \mathbf{y}/\|\mathbf{y}\|_\infty$; for a_i, this is because the mesh fits to the hypercube. We have $(\mathbf{a})_{\uparrow i} - (\mathbf{b})_{\uparrow i} = \frac{\mathbf{a}}{a_i} - \frac{\mathbf{b}}{b_i} = \frac{b_i(\mathbf{a}-\mathbf{b}) - (a_i - b_i)\mathbf{b}}{a_i b_i}$. Thus $\| (\mathbf{a})_{\uparrow i} - (\mathbf{b})_{\uparrow i} \|_\infty \leq \frac{|b_i| \|\mathbf{a}-\mathbf{b}\|_\infty + |b_i - a_i| \|\mathbf{b}\|_\infty}{a_i b_i} \leq \frac{\|\mathbf{b}\|_\infty \|\mathbf{a}-\mathbf{b}\|_\infty + \|\mathbf{b}-\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty}{a_i b_i} \leq \frac{2\|\mathbf{b}-\mathbf{a}\|_\infty}{a_i b_i}$, which is the sought result. To obtain the penultimate inequality, we have used the (already shown) fact that $|b_i| \leq \|\mathbf{b}\|_\infty \leq 1$ along with $|a_i - b_i| \leq \|\mathbf{a}-\mathbf{b}\|_\infty$.

We show (6.17). We have that $(\mathrm{C}(\mathbf{y}))_i = 1/\|\mathbf{y}\|_\infty$, and $(\mathrm{M}_s(\mathbf{y}))_i - (\mathrm{C}(\mathbf{y}))_i \leq s/2$. Hence $\frac{1}{|(\mathrm{M}_s(\mathbf{y}))_i(\mathrm{C}(\mathbf{y}))_i|} \leq \frac{1}{|(\mathrm{C}(\mathbf{y}))_i|-\frac{s}{2}} \frac{1}{|(\mathrm{C}(\mathbf{y}))_i|} \leq 2\|\mathbf{y}\|_\infty^2$ if $\frac{s}{2} < \frac{1}{2}\frac{1}{\|\mathbf{y}\|_\infty}$, i.e., if $s < \frac{1}{\|\mathbf{y}\|_\infty}$. The claim follows.

Finally, (6.18) is direct in view of the above. □

We can now state and prove the announced result.

THEOREM 6.5. Let $\mathring{\mathbf{x}} \in F_{\mathbf{e}_j}$ be a limit of a refining subsequence on face j. Let $\mathbf{v}$ be a refining direction, and let $\{\boldsymbol{\xi}_k\}_{k\in L}$ be a sequence as in Definition 6.5 such that $\lim_{k\in L} \frac{\boldsymbol{\xi}_k}{\|\boldsymbol{\xi}_k\|} = \mathbf{v}$. Assume that there is a constant c_1 such that, for all $k \in L$, $c_1\alpha_k^p \leq \alpha_k^m \|\boldsymbol{\xi}_k\|$. Then $\lim_{k\in L} \frac{\hat{\boldsymbol{\xi}}_k}{\|\hat{\boldsymbol{\xi}}_k\|} = \mathbf{v}$, and thus $\mathbf{v}$ is also an adapted refining direction.

Proof. From Lemma 6.1, one deduces that $\|\alpha_k^m\hat{\boldsymbol{\xi}}_k - \alpha_k^m\boldsymbol{\xi}_k\| \leq 8\underline{\alpha}_k^m$. Since $c_1\alpha_k^p \leq \alpha_k^m\|\boldsymbol{\xi}_k\|$, it follows that $\frac{\|\alpha_k^m\hat{\boldsymbol{\xi}}_k - \alpha_k^m\boldsymbol{\xi}_k\|}{\alpha_k^m\|\boldsymbol{\xi}_k\|} \leq \frac{8\underline{\alpha}_k^m}{c_1\alpha_k^p} \leq \frac{8\alpha_k^m}{c_1\alpha_k^p} = \frac{8}{c_1}\sqrt{\alpha_k^m} \to 0$ as $k \to \infty$, and thus $\lim_{k\in L} \frac{\hat{\boldsymbol{\xi}}_k}{\|\hat{\boldsymbol{\xi}}_k\|} = \lim_{k\in L} \frac{\boldsymbol{\xi}_k}{\|\boldsymbol{\xi}_k\|}$. □

6.5 A Practical SMADS

The original MADS algorithm described in [ACD07] is a general framework in the sense that its convergence properties rely on some conditions on the polling directions. In order to practically implement this algorithm, one has to provide a mechanism to produce admissible polling directions. The first proposed concrete instance of MADS is the Lower Triangular MADS (LTMADS) [ACD07, §4]. We now propose a specific instance of SMADS, inspired from LTMADS. The algorithm, called LTSMADS, is described in Algorithm 6.2. Precisions on the algorithm statement and a convergence analysis follow.

The selection of $\mathbf{x}_{k+1}$ in line 19 of Algorithm 6.2 follows the opportunistic strategy. For simplicity, we do not discuss other strategies (exhaustive, random, ...).

6.5.1 Generating Set of Directions

In the proposed LTSMADS, the directions in $\boldsymbol{\Xi}_k$ respecting the conditions mentioned in Definition 6.1 are generated as in LTMADS [ACD07, §4]. The algorithm is based on a random lower triangular matrix, hence

Algorithm 6.2 : LTSMADS

Refer to Section 6.2 for notation
Input: see Algorithm 6.1
Initialization: see Algorithm 6.1
while $\alpha_k^p > \varepsilon$ **do**
  if $k > 0$ **and** $f(\mathbf{x}_k) < f(\mathbf{x}_{k-1})$ **then**
    SEARCH STEP
    $\mathbf{s}_k := \mathrm{M}_{\underline{\alpha}_{k-1}^m}(\mathbf{x}_{k-1} + 4\alpha_{k-1}^m (\boldsymbol{\xi})_{\hookrightarrow i})$;
    if $f(\mathbf{s}_k) < f(\mathbf{x}_k)$ **then**
      $\mathbf{x}_{k+1} := \mathbf{s}_k$;
    else
      $\mathbf{x}_{k+1} := \mathbf{x}_k$;
    end if
  else
    POLL STEP
    Construct $P_k := \mathrm{M}_{\underline{\alpha}_k^m}(\{\mathbf{x}_k + \alpha_k^m (\boldsymbol{\xi})_{\hookrightarrow i} : \boldsymbol{\xi} \in \boldsymbol{\Xi}_k\})$, where $\boldsymbol{\Xi}_k$ is a positive basis generated as detailed in Section 6.5.1;
    if $f(\mathbf{p}) \geq f(\mathbf{x}_k)$ for all $\mathbf{p} \in P_k$ **then**
      $\mathbf{x}_{k+1} := \mathbf{x}_k$;
    else
      Set $\mathbf{x}_{k+1}$ as the first element $\mathbf{p}$ of P_k such that $f(\mathbf{p}) < f(\mathbf{x}_k)$;
      Set $\boldsymbol{\xi}$ as the corresponding element of $\boldsymbol{\Xi}_k$;
    end if
  end if
  $k := k + 1$;
  Update the parameters α_k^m as in (6.10) and α_k^p as in (6.11);
end while
return $\mathring{\mathbf{x}} := \mathbf{x}_k$;

its name. A brief description of generating $\boldsymbol{\Xi}_k$ is provided below, and a more elaborate treatment is found in [ACD07, §4].

For each iteration k, the lower triangular matrix $\mathbf{L}$ of size $(n-2)\times(n-2)$ is first constructed as a basis in $\mathbb{R}^{n-2}$ with the diagonal elements being either 2^ℓ or -2^ℓ and the lower components being randomly selected from the set $S=\{-2^\ell+1,-2^\ell+2,\ldots,2^\ell-1\}$, where $\ell=-\log_4(\alpha_k^m)$. Next, a specific column $\mathbf{b}(\ell)\in\mathbb{Z}^{n-1}$, relying only on α_k^m, is built such that $|b_{\hat{\imath}}(\ell)|=2^\ell=\frac{1}{\sqrt{\alpha_k^m}}$ for a randomly chosen index $\hat{\imath}\in\{1,\ldots,n-1\}$, and the rest of the components of $\mathbf{b}(\ell)$ are set to be random integers in S. Then a zero vector of length $n-2$ is inserted into $\boldsymbol{L}$ to be the row of index $\hat{\imath}$ in a new matrix $\mathbf{L}'$, of which the remaining rows are permuted. The resulting $(n-1)\times(n-2)$ matrix is augmented by the column $\mathbf{b}(\ell)$ to form a basis $\mathbf{B}$ in $\mathbb{R}^{n-1}$, and the columns of $\mathbf{B}$ are permuted as well to form $\mathbf{B}'$ with evenly distributed zeros. Finally a positive maximal basis denoted as $\boldsymbol{\Xi}_k\in\mathbb{R}^{(n-1)\times 2(n-1)}$ is obtained from $\mathbf{B}'$ by appending to it the negative of each column of $\mathbf{B}'$. This generation process of the poll directions is exemplified below for $n=5$ (where $[S]$ denotes a random number in the set S and $\boldsymbol{B}'_i$ denotes the ith column of the matrix $\boldsymbol{B}'$):

$$\boldsymbol{L}=\begin{pmatrix}\pm 2^l & 0 & 0\\ [S] & \pm 2^l & 0\\ [S] & [S] & \pm 2^l\end{pmatrix}\quad \boldsymbol{b}(l)=\begin{pmatrix}[S]\\ 2^l\\ [S]\\ [S]\end{pmatrix}\leftarrow\hat{\imath}\quad \boldsymbol{L}'=\begin{pmatrix}[S] & \pm 2^l & 0\\ 0 & 0 & 0\\ \pm 2^l & 0 & 0\\ [S] & [S] & \pm 2^l\end{pmatrix}\leftarrow\hat{\imath}$$

$$\boldsymbol{B}=\begin{pmatrix}[S] & \pm 2^l & 0 & b_1\\ 0 & 0 & 0 & b_2\\ \pm 2^l & 0 & 0 & b_3\\ [S] & [S] & \pm 2^l & b_4\end{pmatrix}\quad \boldsymbol{B}'=\begin{pmatrix}0 & b_1 & \pm 2^l & [S]\\ 0 & b_2 & 0 & 0\\ 0 & b_3 & 0 & \pm 2^l\\ \pm 2^l & b_4 & [S] & [S]\end{pmatrix}$$

$$\boldsymbol{\Xi}_k=\begin{pmatrix}\boldsymbol{B}'_1 & \ldots & \boldsymbol{B}'_n & -\boldsymbol{B}'_1 & \ldots & -\boldsymbol{B}'_n\end{pmatrix}.$$

6.5.2 Convergence analysis of LTSMADS

By *b-refining directions* at the limit point $\mathring{\mathbf{x}}$, we mean the refining directions generated from poll directions $\mathbf{b}(\ell)$. Similar to Definition 6.5, we also consider the set of *adapted b-refining directions* constructed from $\hat{\mathbf{b}}(\ell)$ directions. The next result shows that the denseness hypothesis in Theorem 6.4 holds with probability one in LTSMADS.

THEOREM 6.6. Let $\{\mathbf{x}_k\}_{k\in K}$ be a convergent refining subsequence, say on face j, generated by the practical SMADS (Algorithm 6.2). Then the set of adapted b-refining directions associated to K is asymptotically dense in $F_{\mathbf{e}_j}$ with probability one.

Proof. The proof stems from the LTMADS analysis [ACD07, Th. 4.3] and uses Theorem 6.5 for the adapted aspect. Let $\{\mathbf{x}_k\}_{k\in K}$ be a convergent refining subsequence on face j produced by the SMADS, and $\mathring{\mathbf{x}} \in F_{\mathbf{e}_j}$ be the limit of the subsequence. Recall that there is a specific poll direction $\mathbf{b}(\ell)$ among the set of directions $\boldsymbol{\Xi}_k$ generated by the algorithm. The asymptotic denseness theorem [ACD07, Th. 4.3] shows that $\mathbf{b}(\ell)/\left\|\mathbf{b}(\ell)\right\|_\infty$ becomes arbitrarily close to a randomly picked direction $\mathbf{v} \in \mathrm{T}\,\mathrm{aff}(F_{\mathbf{e}_j})$ having $\left\|\mathbf{v}\right\|_\infty = 1$ with a probability

$$P\left[\left\|\frac{\mathbf{b}(\ell)}{\|\mathbf{b}(\ell)\|_\infty} - \mathbf{v}\right\|_\infty < \epsilon\right] \geq \frac{\left(\frac{\epsilon}{4}\right)^{n-2}}{2(n-1)}$$

for any $0 < \epsilon < 1$, provided k is sufficiently large satisfying $\sqrt{\alpha_k^m} = 2^{-\ell} \leq \frac{\epsilon}{2}$. At iterate $\mathbf{x}_k \in F_{\mathbf{e}_j}$, we have an adapted $\mathbf{b}(\ell)$ as defined above; recall that

$$\alpha_k^m \hat{\mathbf{b}}(\ell) := \left(\mathrm{M}_{\underline{\alpha}_k^m}(\mathbf{x}_k + \alpha_k^m \mathbf{b}(\ell))\right)_{\uparrow j} - \mathbf{x}_k.$$

The motivation for the final scaling to $\mathrm{aff}(F_{\mathbf{e}_j})$ is to perform the computations in the chart $(\cdot)_{\uparrow j}$; since the topology of a manifold is defined through the charts, it follows that denseness in a chart is equivalent to denseness on the sphere. We now show that $\frac{\hat{\mathbf{b}}(\ell)}{\|\mathbf{b}(\ell)\|_\infty}$ (which is in the direction of $\hat{\mathbf{b}}(\ell)$) is a sufficiently small perturbation of $\frac{\mathbf{b}(\ell)}{\|\mathbf{b}(\ell)\|_\infty}$ for the gist of the result to be preserved. We have $\|\frac{\hat{\mathbf{b}}(\ell)}{\|\mathbf{b}(\ell)\|_\infty} - \frac{\mathbf{b}(\ell)}{\|\mathbf{b}(\ell)\|_\infty}\|_\infty = \frac{1}{\alpha_k^m\|\mathbf{b}(\ell)\|_\infty}\|\alpha_k^m\hat{\mathbf{b}}(\ell) - \alpha_k^m\mathbf{b}(\ell)\|_\infty = \frac{1}{\alpha_k^m\|\mathbf{b}(\ell)\|_\infty}\|\left(\mathrm{M}_{\underline{\alpha}_k^m}(\mathbf{x}_k + \alpha_k^m\mathbf{b}(\ell))\right)_{\uparrow j} - \mathbf{x}_k - \alpha_k^m\mathbf{b}(\ell)\|_\infty \leq \frac{1}{\alpha_k^m\|\mathbf{b}(\ell)\|_\infty} 8\underline{\alpha}_k^m \leq 2^{3-\ell} \leq 4\epsilon$, where the antepenultimate inequality follows from (6.18) in Lemma 6.1. In conclusion, we

have shown that

$$P\left[\|\frac{\hat{\mathbf{b}}(\ell)}{\|\mathbf{b}(\ell)\|_\infty}-\mathbf{v}\|_\infty<5\epsilon\right]\geq\frac{\left(\frac{\epsilon}{4}\right)^{n-2}}{2(n-1)}\quad\text{for }\sqrt{\alpha_k^m}=2^{-\ell}\leq\frac{\epsilon}{2}. \tag{6.19}$$

Since $\mathbf{b}(\ell)$ is generated independently at each iterate and since, in view of Theorem 6.1, $\sqrt{\alpha_k^m}\leq\frac{\epsilon}{2}$ occurs at infinitely many iterates, it follows that

$$P\left[\|\frac{\hat{\mathbf{b}}(\ell)}{\|\mathbf{b}(\ell)\|_\infty}-\mathbf{v}\|_\infty<5\epsilon\text{ at at least one iterate}\right]=1.$$

This shows asymptotic denseness of the $\hat{\mathbf{b}}(\ell)$ directions. □

6.6 Adapting SMADS for Minimizing Range-based ICA Contrast

Given a random vector $\mathbf{m}\in\mathbb{R}^n$, the proposed ICA scheme estimates an unmixing matrix $\mathbf{X}=\begin{bmatrix}\mathbf{x}^{(1)} & \cdots & \mathbf{x}^{(n)}\end{bmatrix}$ with column vectors $\mathbf{x}^{(j)}\in\mathbb{R}^n$, $j=1,\ldots,n$, subject to the constraint $\left\|\mathbf{x}^{(j)}\right\|_2=1$ such that the n components of $\mathbf{c}=\mathbf{X}^{\mathrm{T}}\mathbf{m}$ are maximally independent as measured by a contrast function. As in [PI96, (4.1)], the contrast function based on a range estimation approach using order statistics is expressed as

$$f(\mathbf{X}):=\sum_{j=1}^{n}\log R\left(\mathbf{x}^{(j)^\top}\mathbf{m}\right)-\log\left|\det\mathbf{X}\right|, \tag{6.20}$$

where $R(Y):=\max(Y)-\min(Y)$ is the range function with Y being a random variable. An estimate of the range of Y is derived in [VLV07] by making use of an ordered finite sequence of observations, $y_{(l)}$, $l=1,\ldots,T$,

$$R(Y):=\frac{1}{h}\sum_{r=1}^{h}R_r(Y) \tag{6.21}$$

with $R_r(Y):=y_{(T-r+1)}-y_{(r)}$. Now we can state the expression for the sample contrast function

$$f(\mathbf{X};\mathbf{M}):=\sum_{j=1}^{n}\log\left(\frac{1}{h}\sum_{r=1}^{h}R_r\left(\mathbf{x}^{(j)^\top}\mathbf{M}\right)\right)-\log\left|\det\mathbf{X}\right|, \tag{6.22}$$

where $\mathbf{X}$ is a candidate unmixing matrix and $\mathbf{M} \in \mathbb{R}^{n \times T}$ is a matrix of T observations of n variables. One possible empirical guideline to set the default value of h, proposed in [VLV07], is

$$h(T) = \max\left(1, \left\lceil \Re\left\{\left(\frac{T-18}{6.5}\right)^{0.65}\right\} - 4.5 \right\rceil\right), \tag{6.23}$$

where $\overline{\rho}$ denotes the nearest integer to ρ and $\Re\{\cdot\}$ returns the real part of the argument. The values of this empirical suggestion for choosing h are represented in Figure 6.5.

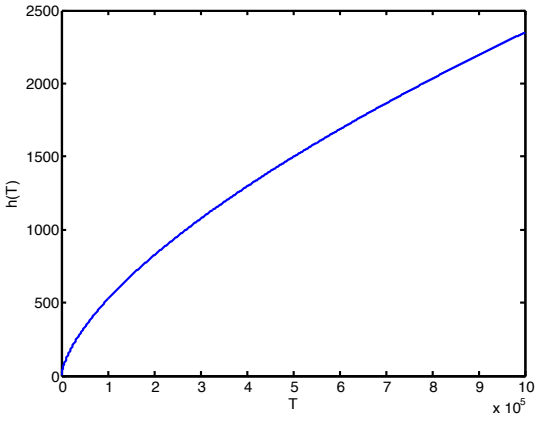


Figure 6.5: Empirical suggestions for setting the values of h in the estimate range function.

Furthermore, the range estimate in (6.21), employing the empirically determined value of h in (6.23), preserves the desirable discriminant contrast property inherent in the exactly evaluated range contrast function [VLV07, p. 816].

We work under the assumption that the data matrix $\mathbf{M}$ is such that the term $\sum_{j=1}^{n} \log\left(\frac{1}{h}\sum_{r=1}^{h} R_r\left(\mathbf{x}^{(j)^\top}\mathbf{M}\right)\right)$ in (6.22) is $> -\infty$ under the constraint $\left\|\mathbf{x}^{(j)}\right\|_2 = 1$. Note that $\sum_{r=1}^{h} R_r\left(\mathbf{x}^{(j)^\top}\mathbf{M}\right) = 0$ if and only if $\mathbf{x}^{(j)^\top}\mathbf{M} = \alpha\mathbf{1}^\top$, where α is a constant. Therefore, we have that

$$\sum_{j=1}^{n} \log\left(\frac{1}{h}\sum_{r=1}^{h} R_r\left(\mathbf{x}^{(j)^\top}\mathbf{M}\right)\right) > -\infty$$

as long as $\mathbf{1} \notin \mathrm{Im}(\mathbf{M}^\top)$. We remark that the assumption $\mathbf{1} \notin \mathrm{Im}(\mathbf{M}^\top)$ does not apply to ICA problems where the original sources are believed to be constant signals/images, i.e., their distributions are degenerate. In practice, the situation $\mathbf{1} \in \mathrm{Im}(\mathbf{M}^\top)$ will not usually arise, because the condition $\mathbf{1} \in \mathrm{Im}(\mathbf{M}^\top)$ means that all the T observations in $\mathbb{R}^n$ belong to a same hyperplane. As soon as $T \geq n$, this situation occurs with zero probability under reasonable assumptions on the observation noise.

The following lemma guarantees that the Lipschitz assumption frequently made in the above convergence analysis of (LT)SMADS holds for the contrast function (6.22).

LEMMA 6.2. The range-based ICA contrast function f (6.22) is Lipschitz near every $\mathbf{X}$ such that $-\infty < f(\mathbf{X}) < \infty$.

Proof. The range-based contrast function in (6.22) involves the sum of max functions. Since the max function is Lipschitz and the log function is differentiable on $(0, \infty)$, it follows that the term

$$\sum_{j=1}^{n} \log \left(\frac{1}{h} \sum_{r=1}^{h} R_r \left({\mathbf{x}^{(j)}}^\top \mathbf{M} \right) \right)$$

is Lipschitz near $\mathbf{x}$ if and only if $\sum_{r=1}^{h} R_r \left({\mathbf{x}^{(j)}}^\top \mathbf{M} \right) \neq 0$, $j = 1, \ldots, n$. Besides, the barrier term $-\log|\det \mathbf{X}|$ is Lipschitz near every $\mathbf{X}$ where it is finite. □

Our objective function is thus the range-based ICA contrast in (6.22) defined on the set of $n \times n$ matrices whose columns have unit Euclidean norm, denoted here by

$$\mathcal{OB}(n) = \left\{ \mathbf{X} \in \mathbb{R}^{n \times n} : \mathrm{ddiag}(\mathbf{X}^\top \mathbf{X}) = \mathbf{I}_n \right\}, \tag{6.24}$$

where $\mathrm{ddiag}(\cdot)$ represents the diagonal matrix whose diagonal elements are those of the matrix in the argument, and $\mathbf{I}_n$ is the $n \times n$ identity matrix. Since $\mathcal{OB}(n)$ is merely the Cartesian product of n copies of $\mathbb{S}^{n-1}$, the generalization of SMADS (Algorithm 6.1) to $\mathcal{OB}(n)$ is straightforward.

We point out that our definition of $\mathcal{OB}$ in (6.24) does not exactly coincide with its definition given in [TL02], as it does not impose that its elements have full rank. The distinction is irrelevant for the objective

function (6.22) since the $-\log|\det \mathbf{X}|$ term guarantees that the iterates of any descent iteration stay away from the set of rank-deficient matrices.

What follows is a theorem that justifies the choice of the SMADS algorithm to seek for a local minimum of the contrast function defined in (6.22).

THEOREM 6.7. For the range-based ICA contrast function f in (6.22), the sequence generated by the SMADS algorithm (Algorithm 6.2) admits at least one convergent refining subsequence, and the limit point of any such subsequence is a Clarke stationary point of f.

Proof. The result follows from compactness of $\mathcal{OB}$ and from Theorem 6.4. □

REMARK 6.1. In our numerical experiments, we have always observed that the *whole* SMADS sequence converges, and it then follows from Theorem 6.7 that it converges to a Clarke stationary point. Moreover, since SMADS is a descent method, convergence to points that are not local minimizers is unstable under perturbations, and is not expected to occur in practice.

6.7 Experimental Results

6.7.1 Simulation with Various Image Datasets

The range-based contrast function assumes that the source support measure is finite, in other words, that the underlying sources are bounded. This assumption holds good in the case of images. Moreover, the image sources are in general not uncorrelated and their density distributions are usually multimodal, which exacerbate the challenges faced by an ICA technique. Furthermore the earlier works using the range-based contrast either recommend the use of image data [VLV07] or validate the algorithm with face images [Vri07]. Therefore we have considered the following image categories for assessing the performance of our approach: (i) two sets of 12 natural images including sceneries, human portraits, man-made structures, animals and birds from the Berkeley segmentation dataset and benchmark [MFTM01]; (ii) nine images under each category of aerial and texture images acquired from the Signal and Image Processing Institute, University of Southern California at "http://sipi.usc.edu/database"; (iii) nine face images from the dataset of

Informatics and Mathematical Modelling, Technical University of Denmark [SEL03]. All the test images were resized to have 200×200 pixels each and converted into gray-scale images. From each dataset comprising natural, aerial, texture and face images, 25 random combinations of six images were generated to assess the range-based ICA on $\mathcal{OB}$ in terms of the source separation quality. During each trial, the multidimensional data of size 40000×6 consisting of column-wise concatenated pixel gray-levels was mixed by a mixing matrix built from 36 coefficients drawn from a uniform distribution on the interval $(0, 1)$. It is remarked that the mixing procedure is advocated in the works of [KOW$^+$97, p. 497] concerning the source separation of multidimensional image data. The resulting mixture was whitened, because pre-whitening is deemed as the standard preprocessing procedure [LA10] and it is believed to improve the convergence, although strictly speaking, ICA algorithms on $\mathcal{OB}$ obviate the need for pre-whitening. The mixture/pre-whitened data thus obtained was supplied as the input for the below-mentioned ICA algorithms for a relative assessment of the source separation performance:

- fastICA [Hyv99a], JADE [CS93], and infomax [MBJS96], which are widely reported in the ICA literature;
- SWICA [VLV07], where the range-based function in (6.22) was first introduced;
- NOSWICA [LVV06] meant for the separation of highly correlated sources;
- ICA by entropy bound minimization (ICA-EBM) [LA10] that uses an accurate entropy estimator and adopts a line-search optimization procedure;
- ICA by optimizing quadratic measures of independence (QICA) with an exhaustive search strategy [SRPP11] in the quest of avoiding local optima;
- our proposed LTSMADS (Algorithm 6.2) on the $\mathcal{OB}$ manifold (OBMADS) minimizing the range-based function in (6.22).

It is pointed out that the image mixture is generated in an artificial manner to enable us to compare the unmixing matrix/source estimate with the true unmixing matrix/source. Further research is underway,

where the algorithm can be potentially employed to separate signal mixtures arising from real scenarios, e.g., unmixing electroencephalogram recordings from scalp electrodes prior to generating brain maps to help diagnose focal epilepsy seizures. In such real scenarios, since the true unmixing matrix/source are unknown, one must resort to indirect quality assessment methods in order to compare the various ICA algorithms; for example, in electroencephalogram experiments that we will report elsewhere, an expert neurologist evaluates the accuracy with which the brain maps generated from the ICA output matches the patient's pathological condition.

The parameter values for the OBMADS algorithm were set as prescribed in [ACD07]: $\tau = 4$, $\beta^+ = 1$, $\beta^- = -1$, $\varepsilon = 10^{-10}$, $\alpha_0^m = \alpha_0^p = 1$, and total function evaluations $f_{\mathrm{e}_{\max}} = 10^5$. The ICA unmixing matrices yielded by the methods included in the empirical study were used to reconstruct the sources from the mixtures. Since the ambiguity of sign and the indeterminacy of the order of the ICs hold for the ICA model, the source estimates are necessarily rescaled and reordered prior to computing the root-mean-square error (RMSE) values with respect to the original sources. Given the original vectorized images $\mathbf{a}^{(j)} = [a_1^{(j)}, \ldots, a_T^{(j)}]$, $j = 1, \ldots, n$, and the estimated ones after reordering $\mathbf{c}^{(j)} = \mathbf{x}^{(j)^\top}\mathbf{M} = [c_1^{(j)}, \ldots, c_T^{(j)}]$, $j = 1, \ldots, n$, the RMSE is computed as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{j=1}^{n}\sum_{l=1}^{T}\left(a_l^{(j)} - c_l^{(j)}\right)^2}{\sum_{j=1}^{n}\sum_{l=1}^{T}\left(a_l^{(j)}\right)^2}}. \tag{6.25}$$

The RMSE measure in (6.25) is always nonnegative, scale invariant, and it circumvents the permutation ambiguity due to the reordering of the source estimates.

The mean and standard deviation of the RMSE values in 25 trial runs of the experimented schemes with a face, aerial, texture dataset containing nine images each, and two datasets comprising 12 natural images each are reported in Table 6.1. Though the fastICA that maximizes the negentropy of the estimated sources with an approximate Newton method offers computational advantage, the mean RMSE is high. The JADE algorithm aims at minimizing the sum of the squared cross-cumulants of the source estimates; the tensorial algorithm faces limitations in higher dimensions as evidenced by the mean RMSE. The

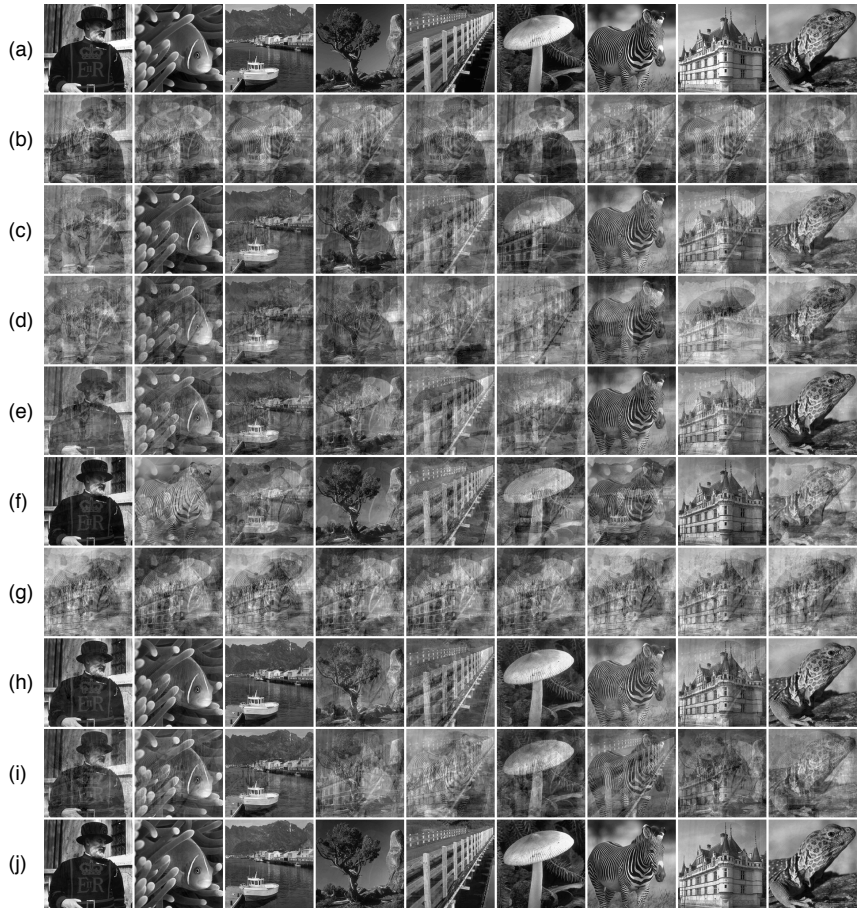


Figure 6.6: Simulation results with nine natural images of size 200×200 for subjective assessment. (a) Original source images. (b) Mixed images. (c)–(j) Estimated and reordered sources from (c) fastICA, (d) JADE, (e) infomax, (f) SWICA, (g) NOSWICA, (h) ICA-EBM, (i) QICA and (j) OBMADS with the RMSE values of 0.265, 0.353, 0.267, 0.248, 0.396, 0.127, 0.268 and 0.039, respectively.

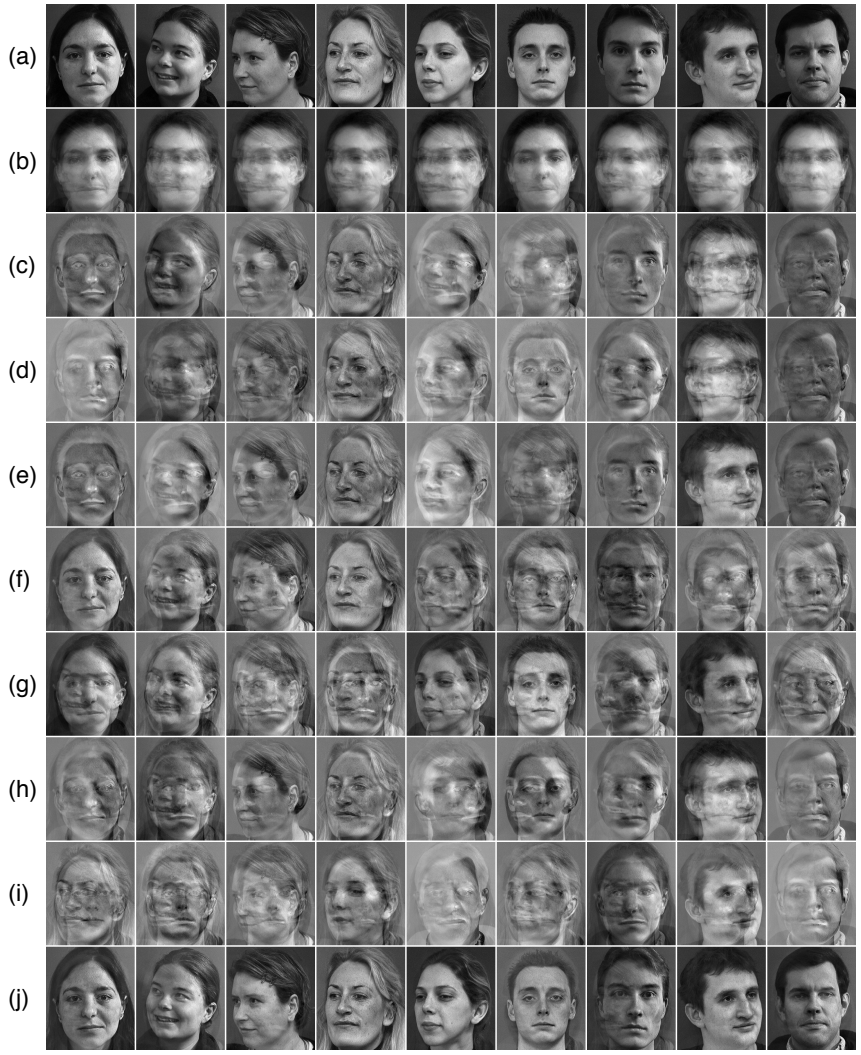


Figure 6.7: Unmixing results using nine face images of size 350×300 for visual scrutiny. (a) Original source images. (b) Mixed images. (c)–(j) Estimated and reordered sources from (c) fastICA, (d) JADE, (e) infomax, (f) SWICA, (g) NOSWICA, (h) ICA-EBM, (i) QICA and (j) OBMADS with the RMSE values of 0.458, 0.466, 0.468, 0.375, 0.365, 0.451, 0.467 and 0.114, respectively.

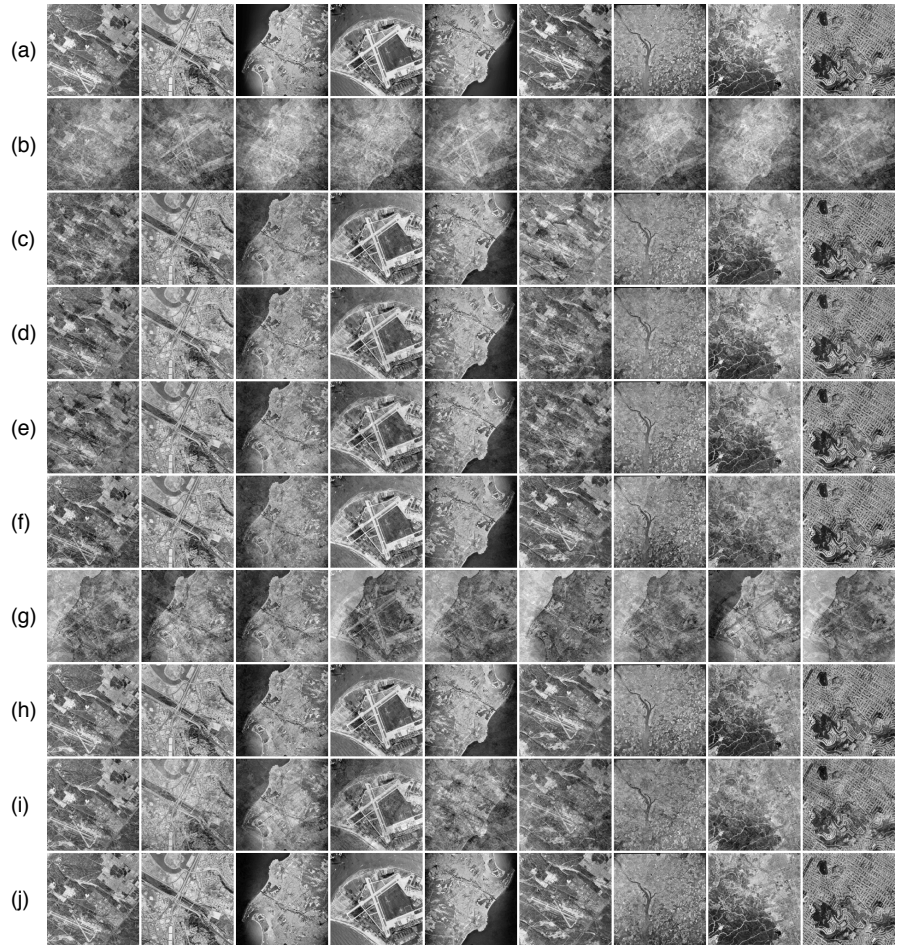


Figure 6.8: Unmixing results using nine aerial images for visual scrutiny. (a) Original source images. (b) Mixed images. (c)–(j) Estimated and reordered sources from (c) fastICA, (d) JADE, (e) infomax, (f) SWICA, (g) NOSWICA, (h) ICA-EBM, (i) QICA and (j) OBMADS.

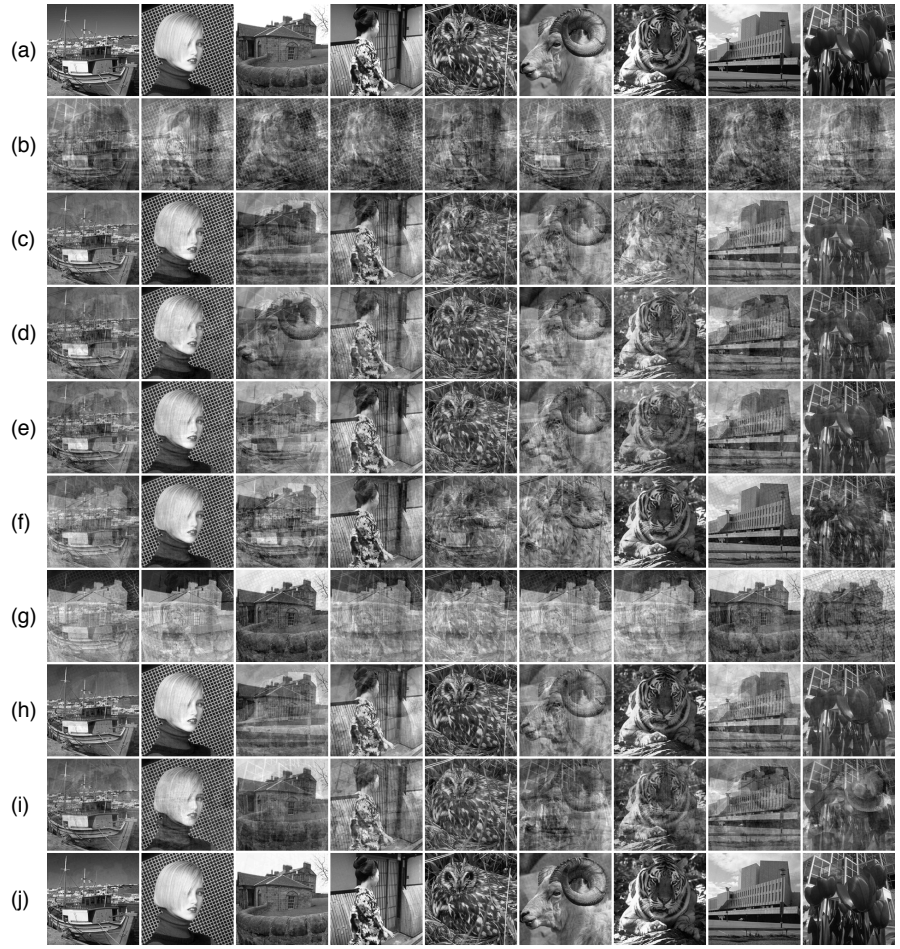


Figure 6.9: Unmixing results using nine natural images for visual scrutiny. (a) Original source images. (b) Mixed images. (c)–(j) Estimated and reordered sources from (c) fastICA, (d) JADE, (e) infomax, (f) SWICA, (g) NOSWICA, (h) ICA-EBM, (i) QICA and (j) OBMADS.

Table 6.1: Mean and standard deviation of RMSE values from the investigated ICA schemes with a face, aerial, texture and natural image dataset. This empirical study involves 25 trial runs with the mixture of six randomly selected 200×200 images from each dataset. The face, aerial and texture dataset comprises nine images each, and two natural image datasets contain 12 images each. The values in bold face represent the minimum obtained among the experimented schemes.

ICA	RMSE mean and std. dev.				
algorithm	natural I	natural II	face	aerial	texture
fastICA	0.169 ± 0.054	0.254 ± 0.050	0.439 ± 0.025	0.109 ± 0.026	0.018 ± 0.002
JADE	0.143 ± 0.037	0.230 ± 0.055	0.435 ± 0.031	0.092 ± 0.024	0.017 ± 0.002
infomax	0.158 ± 0.050	0.257 ± 0.052	0.423 ± 0.033	0.108 ± 0.031	0.018 ± 0.002
SWICA	0.138 ± 0.051	0.196 ± 0.045	0.335 ± 0.036	0.095 ± 0.026	0.099 ± 0.042
NOSWICA	0.297 ± 0.131	0.321 ± 0.129	0.356 ± 0.101	0.211 ± 0.127	0.237 ± 0.108
ICA-EBM	0.100 ± 0.024	0.113 ± 0.066	0.410 ± 0.045	0.058 ± 0.016	0.017 ± 0.002
QICA	0.182 ± 0.052	0.192 ± 0.053	0.398 ± 0.043	0.115 ± 0.038	0.091 ± 0.035
OBMADS	$\mathbf{0.034 \pm 0.008}$	$\mathbf{0.040 \pm 0.036}$	$\mathbf{0.086 \pm 0.011}$	$\mathbf{0.042 \pm 0.013}$	$\mathbf{0.008 \pm 0.001}$

infomax employs a stochastic gradient ascent rule to maximize the entropy of the ICs that are nonlinearly transformed; as expected, the surrogate contrast resulted in an inferior outcome.

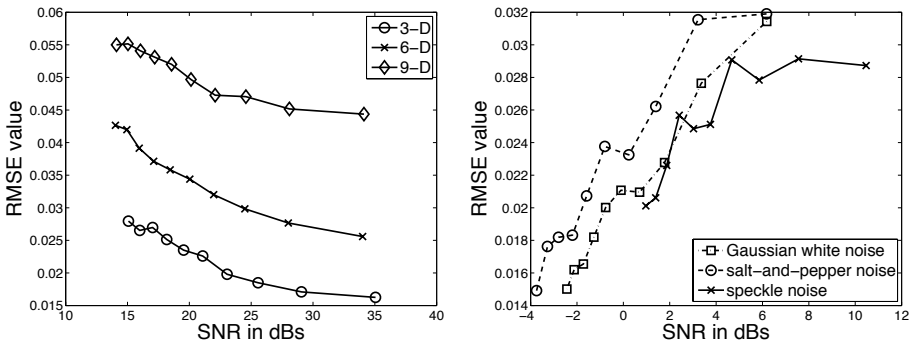
The SWICA, NOSWICA and OBMADS minimize the same contrast function in (6.22). While the SWICA insists on the orthogonality constraint with the Givens rotation matrices, the NOSWICA intentionally relaxes this constraint in estimating the ICs which hopefully represent the correlated sources. The support-width ICA methods have been claimed to enjoy the following advantages [VLV07]: (i) an improvement in the separation performance; (ii) applicability in situations where sources may be correlated; (iii) ability to recover the sources with strongly bimodal densities; (iv) robustness to the dimensionality of the source space. The emphasis in [VLV07] is a systematic study of the range-based contrast function, and the nonsmooth optimization recommended in the SWICA is admittedly simple. The NOSWICA motivates further research in the direction of allowing more degrees of freedom in the ICA estimation by taking into account the correlatedness assumption among the sources constructed from real-world data. It has been remarked in [Vri07, p. 262] that the robustness of NOSWICA is disappointing as the relaxation of orthogonality constraint comes with a price

of a larger space of solutions, which may account for the variability in results. Indeed the aforestated reasons underpin the development of a robust nonsmooth optimization algorithm on the $\mathcal{OB}$ manifold that can efficiently handle the range-based function as corroborated by the mean RMSE values listed in Table 6.1.

Since we intend to compare our approach with some recent ICA algorithms, the ICA-EBM and QICA were incorporated into the empirical study. The QICA offers a unified framework for a number of independence measures, and it generalizes the concept of information theoretic learning (ITL); the similarity between two random variables is evaluated by the inner product of their probability density functions. Since the QICA stresses the accurate estimation of ICs rather than speeding up the estimation, we find it appropriate for the comparative study. The ICA-EBM relies on accurately estimating the entropy of a random variable with the help of the maximum entropy bound, given the observations. The method is claimed to be effective in separating the sources following a wide range of distributions. The fact that it is more common to encounter not-so-trivial distributions of the image pixel gray-levels justifies the inclusion of the ICA-EBM in simulations. Inspite of increased computational load, the QICA gave rise to a large mean RMSE. On the other hand, the source estimates of acceptable quality were obtained in the ICA-EBM.

Noteworthily, the OBMADS is capable of estimating the ICs more accurately, and the reduction in the mean RMSE of the source estimates can be ascribed to the efficiency of the Riemannian nonsmooth optimizer which ensures convergence to a Clarke stationary point. Furthermore, the Wilcoxon's signed-rank test has been carried out to affirm that the RMSE reduction attained by OBMADS in unmixing the sources from all the experimented image datasets is statistically significant (at the 0.05 significance level) compared to the competing approaches. Except in three[2] out of 35 instances, the p-value obtained is 1.2290×10^{-05}, which implies that the RMSE from OBMADS is less than that of the method under comparison in all the 25 trial runs. It is remarked that in the Wilcoxon's signed-rank test, when a random sample of size z is element-

[2] p-values corresponding to the ICA-EBM versus OBMADS in the aerial and natural II dataset are 1.3898×10^{-05} and 3.6243×10^{-05}, respectively, as OBMADS outclassed the ICA-EBM in 24 trial runs in the former and in 23 in the latter case. Similarly, in the natural II dataset, the SWICA versus OBMADS resulted in a p-value of 1.3898×10^{-05}.



(a) (b)

Figure 6.10: Effect of noise perturbation on the unmixing performance of OBICA measured by the RMSE value, when the image sources are corrupted using (a) unbounded AWGN; (b) bounded noise models (`imnoise` command in Matlab) prescribed for images. The signal-to-noise-ratio (SNR) is specified in decibels (dBs).

wise strictly less (or more) than the other sample, then the statistic assumes a fixed value given by $z(z+1)/2$ regardless of the actual values of the samples. As a consequence, the p-value will remain the same for the comparisons wherein the investigated method outperforms the competitors in a fixed number of trial runs. The Wilcoxon's signed-rank test is more meaningful here, since it belongs to the category of nonparametric hypothesis testing methods, where the assumption that two compared populations are normally distributed does not necessarily hold.

Finally, to bear evidence subjectively for the appealing performance of the proposed optimizer coupled with a range-based contrast function, two instances of the natural and face image simulation results from all the investigated ICA algorithms are portrayed in Figures 6.6 and 6.7 along with the respective RMSE values. More visual results are presented in Figures 6.8 and 6.9.

6.7.2 Discussion on OBMADS Behavior

6.7.2.1 Noise Perturbation

To investigate the robustness of OBMADS under noise perturbations, the vector of concatenated pixel gray-levels from a noisy image is first considered as an unbounded source, i.e., the data-points are not necessarily confined to the interval $[0, 255]$. A linear combination of three, six or nine natural images contaminated with the additive white Gaussian noise (AWGN), having zero mean and the standard deviation incrementally selected in the interval [1, 10], was supplied as the input for the OBMADS algorithm. As can be envisaged, the unmixing performance measured by the RMSE degrades as shown in Figure 6.10(a). Nevertheless, in the aforementioned setting, the boundedness assumption to be valid for the sources in a range-based ICA model is clearly violated. Therefore an interesting observation is included in Figure 6.10(b) for a set of six natural images, while various types of noise—AWGN, salt-and-pepper noise and multiplicative speckle noise—commonly used by the image processing community were introduced in an artificial manner[3]. When the noisy pixel values are bounded, the situation turns out to be in favor of our approach, since a consequence of noise addition is an increase in the number of pixels with extreme gray-levels. Apparently the pixels whose gray-levels are either zero or 255 facilitate any range-based ICA algorithm to estimate the range of the unmixed sources more accurately.

6.7.2.2 Ill-conditioned Mixing Matrix

To examine whether the performance of OBMADS relies on the condition number of the mixing matrix, mixtures of three, six or nine natural images generated using mixing matrices with a wide-range of condition numbers were allowed to be source-separated. The RMSEs between the true and estimated sources that correspond to different condition numbers of the mixing matrices were recorded. The plot in Figure 6.11 enables us to conclude that the separation performance of OBMADS is not adversely affected by the ill-conditioned mixing matrix.

[3]The Matlab command `imnoise` adds a specific type of noise to an image such that the noisy image pixel values are bounded in the interval $[0, 255]$.

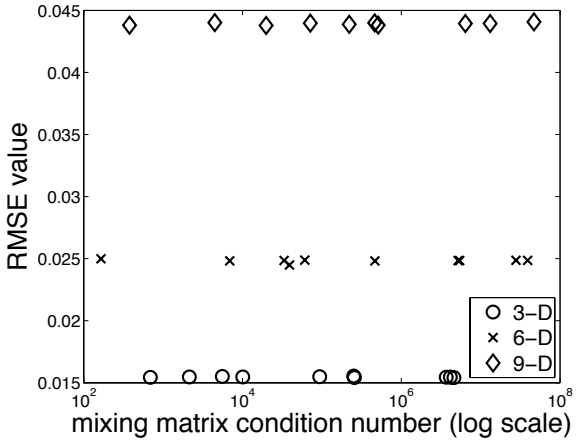


Figure 6.11: The plot of the RMSE between the true and estimated sources versus the condition number of the mixing matrices suggests that the performance of OBMADS is not influenced by an ill-conditioned mixing matrix.

6.7.2.3 Source Correlatedness

As the proposed optimization relaxes the orthogonality constraint and remains on $\mathcal{OB}$, it is ideally suited for unmixing the sources which are not uncorrelated. Importantly, it is not uncommon to come across sources in practice which do not comply with the strict uncorrelatedness assumption imposed by several ICA algorithms. For instance, the face or landscape images share a common underlying pattern and tend to be highly correlated [LVV06]. In such circumstances, as emphatically stated in [SAQ$^+$12], the whitening process does not restrict the search for the unmixing matrix to the space of orthogonal matrices. To visualize the degree of correlation amongst the images in each dataset (face, natural, aerial and texture), a pictorial representation of the correlation matrices is provided in Figure 6.12. A closer scrutiny would reveal that the face (Figure 6.12(a)) or natural images (Figures 6.12(b) and 6.12(c)) are more correlated than the aerial (Figure 6.12(d)) or texture images (Figure 6.12(e)). The efficacy of the proposed $\mathcal{OB}$ optimization algorithm can be endorsed by the illustrations in Figures 6.6 and 6.7, and the relatively small RMSE values yielded by OBMADS in comparison with the state-of-the-art approaches for face and natural image sources as consolidated in Table 6.1.

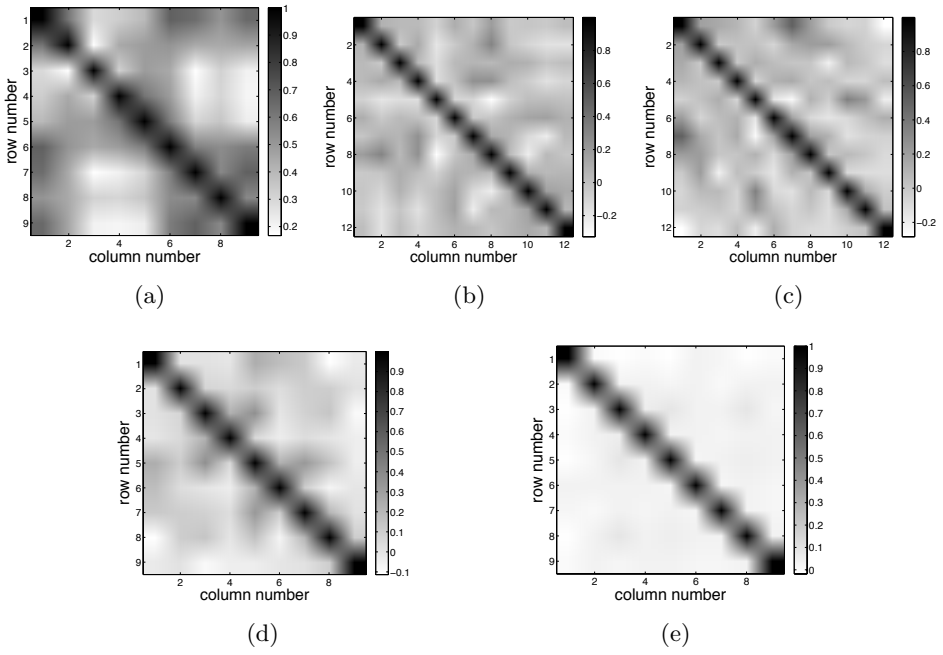


Figure 6.12: Correlation matrices generated with the source images are displayed for the following datasets: (a) face; (b) natural I; (c) natural II; (d) aerial; (e) texture.

Table 6.2: Mean and standard deviation of the computational time (in seconds) taken by various ICA methods in ten trial executions. In each run, all the algorithms were supplied with the same mixture generated from 6−, 7−, 8− or 9−D natural image data of size 50×50. This table should be considered together with Figure 6.6, which shows that OBMADS noticeably dominates the other methods in terms of unmixing quality.

ICA algorithm	mean and std. dev. of computational time (seconds)			
	6−D	7−D	8−D	9−D
fastICA	0.283 ± 0.522	0.322 ± 0.507	0.297 ± 0.579	0.296 ± 0.673
JADE	0.012 ± 0.006	0.016 ± 0.006	0.019 ± 0.007	0.028 ± 0.007
infomax	2.118 ± 0.310	2.309 ± 0.266	2.727 ± 0.611	3.659 ± 1.356
SWICA	1.032 ± 0.009	1.534 ± 0.014	2.147 ± 0.015	3.481 ± 0.034
NOSWICA	0.920 ± 0.175	1.304 ± 0.243	1.772 ± 0.253	2.579 ± 0.345
ICA-EBM	0.308 ± 0.053	0.361 ± 0.083	0.553 ± 0.117	0.600 ± 0.119
QICA	32.373 ± 0.287	45.365 ± 0.370	59.963 ± 0.901	77.398 ± 1.021
OBMADS	19.072 ± 1.161	31.391 ± 0.288	39.097 ± 0.092	49.711 ± 0.383
CE-vMF	32.307 ± 5.943	62.567 ± 9.572	105.657 ± 15.501	196.412 ± 29.310

6.7.2.4 Computational Cost

The software system was implemented in Matlab R2011b on a MacBook Pro (Intel Core i7 2.2 GHz CPU, 8 GB 1333 MHz DDR3) using Mac OS X Lion 10.7.4.

In Figure 6.6, the proposed OBMADS is the only tested method that arguably produces an unmixing of sufficient quality. Its computational time was of the order of one minute, which makes it adequate for offline source separation tasks. For information, timing comparisons are furnished in Table 6.2. The table also features the CE-vMF method of [SCAA12], which produces results of similar quality as OBMADS but is seen to be slower.

6.8 Conclusion

We have proposed and analyzed an extension to the sphere of the MADS algorithm of [ACD07]. A specific instance, called LTSMADS, of the re-

sulting derivative-free algorithmic framework has been proposed and extended to an optimization algorithm, called OBMADS, evolving on the set of matrices with unit-norm column constraint. OBMADS has then been applied to a range-based contrast function for ICA due to Pham, Vrins and collaborators. An analysis has been carried out to show that OBMADS converges to Clarke stationary points of the contrast function. Since the contrast function has the discriminacy property, it follows that bounded and independent sources are recovered in practice from their linear mixtures by OBMADS in the infinite-sample limit. The efficacy of OBMADS when the unmixing matrix is estimated from a mixture of quasi-uncorrelated sources has been demonstrated using simulations with natural, face, aerial and texture images. Though the chapter focuses on an ICA application, the proposed optimization algorithms are applicable to several potential problems in science where one has to optimize a locally Lipschitz continuous function subject to unit-norm constraints.

Chapter 7

Economic load dispatch with valve-point effect

Chapter overview

The economic load dispatch problem (ELDP) is a classical problem in the power systems community. It consists in the optimal scheduling of the output of power generating units to meet the required load demand subject to unit and system inequality and equality constraints. This optimization problem is challenging on three different levels: the geometry of its admissible set, the non-differentiability of its cost function and the multimodal aspect of its landscape. For this reason, ELDP has received much attention in the past few years and numerous derivative-free techniques have been proposed to tackle its multimodal and non-differentiable characteristics. In this chapter, we propose a different approach exploiting the rich geometrical structure of the problem. We show that the non-linear equality constraint can be handled in the framework of Riemannian manifolds and we develop a projected subgradient descent algorithm to provide fast and robust convergence to local minima, producing admissible iterates at all times. To this end, we show that Clarke's calculus can be used to compute a deterministic admissible descent direction by solving a simple, low-dimensional quadratic program. Finally, we show that the proposed algorithm, being a local optimization method, can be incorporated in existing heuristic techniques to provide better exploration of the search space.

The material of this chapter is based on the following publication:
[BSBA13] Pierre B. Borckmans, S. Easter Selvan, Nicolas Boumal, and P.-A. Absil. A Riemannian Subgradient Algorithm for Economic Dispatch with Valve-Point Effect. *J. Comput. Applied. Math.*, 2013, accepted.

7.1 Introduction

The economic load dispatch problem (ELDP) is the optimal scheduling of the output of power generating units to meet the required load demand subject to unit and system equality and inequality constraints [SCC03]. In the traditional ELDP, the cost function for each generator is modelled by a single quadratic function. Nevertheless, in practice, one has to take into account highly nonlinear input–output characteristics arising due to valve-point loadings or generating unit ramp rate limits. As a consequence, we end up with a nonsmooth, equality- and inequality-constrained optimization problem, (7.5), which is in general multimodal (it presents several local optima) and for which classical smooth optimization techniques are thus not suitable.

For this reason, the ELDP has received much attention in the past few years and numerous derivative-free techniques have been proposed to tackle its multimodal and nondifferentiable aspects. Popular techniques include genetic algorithms (GA) [WS93], evolutionary programming [YYH96], particle swarm optimisation (PSO) [PLSL05], and differential evolution (DE) [NI08].

More recently, "global–local" hybrid methods have appeared that combine a "global" method having good global searching abilities and a "local" method with better fine-tuning abilities. Even though there may not be a clear frontier between the two groups of methods, it is quite evident that methods such as GA, PSO and DE belong to the "global" group whereas a method such as sequential quadratic programming (SQP) belongs to the "local" group. Desirable properties of a local method include the following. (i) The method should be *feasible*, i.e., all the iterates should satisfy the equality and inequality constraints. Indeed, a drawback of infeasible methods is that even if the outcome is close to satisfying the constraints, the cost function value at the outcome may be quite different from its value at the nearest point satisfying the constraints. Another downside of infeasibility is that the intermediate results are not directly usable since they do not respect the balance con-

straint. (ii) The iterates should converge to a local minimizer of the cost function. (iii) There is a well-known trade-off between low numerical cost of the iteration and fast convergence of the iterates; as a simplistic illustration, concatenating a few steps of a given method improves the speed of convergence per iterate but degrades the numerical cost per iterate. If the sought accuracy is sufficiently low—which is arguably the case of the ELDP where the coefficient values and even the model itself are debatable—then it is preferable to run a lot of cheaper iterations, as this makes it possible to check a stopping criterion more frequently and hence to avoid the unnecessary computational effort inherent to over-convergence. (iv) Methods that exploit the very particular structure of the ELDP should be preferred over "out of the box" numerical algorithms.

SQP is a popular choice for the local method in global–local ELDP algorithms; see, e.g., [VJ04, AKTH02, CLL+12, ASAO10]. Whereas SQP is a welcome complement to global search methods, it does not satisfy any of the properties mentioned above. Specifically, it is an infeasible method, since the equality constraint is only satisfied in the limit. Convergence, while empirically observed, is not guaranteed because the classical convergence theory of SQP assumes, among other things, that the cost function and the constraint functions have continuous first derivatives [NW06, theorem 18.3], which is not the case in the ELDP (7.5). As a quadratically convergent method [NW06, theorem 18.4], it can be seen as favouring fast convergence of the iterates at the expense of a higher numerical cost per iterate. Finally, it is a general-purpose algorithm, not specifically tailored to the ELDP.

Other local methods have been proposed in combination with global methods. The Nelder-Mead (NM) method has been shown to be an effective local method in combination with PSO [Nik10], but it does not exploit the readily accessible first-order information on the cost function. The modified subgradient (MSG) method used in [YÖ11] exploits first-order information on the cost function, but feasibility of the iterates is achieved only after a certain number of steps in view of the *sharp* augmented Lagrangian approach. Shor's r-algorithm, combined with an improved differential evolution (IDE) method in [YWZY09], is also of the subgradient type, but the iterates are not feasible.

In this chapter, we develop a novel method that satisfies properties (i), (iii), and (iv). It is also built to satisfy (ii) and does it empirically, however, as a consequence of (iv), its convergence does not immediately

follow from an existing result, and a detailed convergence analysis is beyond the scope of this chapter.

Property (i), feasibility of the iterates, is enforced using the framework of Riemannian optimization. The adequacy of this framework stems from the fact that the equality constraint (7.3) defines an ellipsoid, which admits a natural structure of a Riemannian manifold. Riemannian optimization, also called optimization on manifolds, is nowadays a vibrant area of research, whose foundations can be found, e.g., in [HM94, EAS98, AMS08]. Since this framework is new in the ELDP context, we dedicate a significant part of this chapter to laying out the necessary background.

Property (iii) comes by preferring a steepest-descent approach over a second-order approach. However, in view of the non-smoothness due to the valve-point effect, the steepest-descent approach does not rely on gradient techniques but rather on subgradient techniques, using the framework of Clarke's generalized calculus [Cla75].

In view of the above, the proposed method fits in the framework of Riemannian subgradient descent. With respect to the general-purpose Riemannian subgradient descent method of Dirr et al. [DHL07], a contribution of our development is to incorporate bound constraints in order to handle the generator capacity constraints (7.2) present in the ELDP. Another contribution of this work is that, whereas many heuristic algorithms for the ELDP consist of (a combination of) existing black-box optimization techniques, the proposed method strives to exploit as much as possible the very particular structure of the ELDP. This allows notably for an efficient representation of the generalized gradient, which enables fast computation of a descent direction by solving a low-dimensional quadratic program.

In summary, the proposed technique provides fast convergence to a nearby local minimum of the ELDP (7.5), while satisfying the power balance (7.3) and capacity constraints (7.2) throughout the optimization process. Therefore, the aforementioned heuristics largely explored in the literature and the proposed subgradient descent algorithm present very complementary properties: the multimodal aspect of the ELDP can be addressed using any global feasible exploration tool while the local refinement of a potential solution is efficiently provided by the proposed approach, including a check for the stationarity of the final iterate.

The remainder of the chapter is organized as follows. In Section 7.2, the ELDP with the valve-point effect is briefly presented, followed by a

detailed treatment of the underlying geometry of the optimization problem and an introduction to the necessary differential geometry tools in Section 7.3. The subgradient descent algorithm is presented in Section 7.4. Subsequently, its formulation on the Riemannian manifold and its specialization for the ELDP are discussed. Section 7.5 presents the implementation details. In Section 7.6, numerical results are presented and we show how our algorithm can be hybridized with a global scheme to deal with the multimodal aspect of the ELDP. Finally, conclusions are drawn in Section 7.7.

7.2 ELDP Considering Valve-point Effect

7.2.1 Problem Statement

Traditionally, the generating-unit cost functions are considered to be convex with the heat rate curves exhibiting monotonically increasing characteristics. However, in reality, the steam admission valves in the large steam turbines cause discontinuities in the incremental heat rate curves. Therefore, to accurately model the ELDP, the valve-point loadings in the n generating units are to be incorporated, leading to nonconvex input-output characteristics of the generating units [VJ05]; the cost function is then stated as

$$\begin{aligned} f_{\mathrm{T}}(\boldsymbol{p}) &= \sum_{i=1}^{n} f^i(p_i), \qquad \boldsymbol{p} = [p_1, p_2, \ldots, p_n]^\top \\ &= \sum_{i=1}^{n} a_i p_i^2 + b_i p_i + c_i + \left| d_i \sin\left[e_i \left(p_i^{\min} - p_i\right)\right]\right|. \end{aligned} \tag{7.1}$$

Here $f_{\mathrm{T}}(\boldsymbol{p})$ is the total production cost (\$/h) pertaining to the n-dimensional output power vector $\boldsymbol{p}$ and $f^i(p_i)$ is the incremental fuel cost function (\$/h) pertaining to the real power output of the ith unit, p_i. For the ith generating unit, the cost coefficients are denoted by a_i, b_i, c_i, and the constants from the valve-point effect by d_i, e_i. All the coefficients are positive. This formulation of the EDLP is static and the coefficients are thus constants. In matrix notation, we have

$$f_{\mathrm{T}}(\boldsymbol{p}) = \boldsymbol{p}^\top \mathrm{Diag}(\boldsymbol{a})\boldsymbol{p} + \boldsymbol{b}^\top \boldsymbol{p} + \boldsymbol{c}^\top \mathbf{1} + \boldsymbol{d}^\top \left|\sin\left[\boldsymbol{e} \circ (\boldsymbol{p}^{\min} - \boldsymbol{p})\right]\right|,$$

where $\circ$ denotes the component-wise product, $\mathrm{Diag}(\,\cdot\,)$ denotes the diagonal matrix obtained from the entries of its vector argument, and the sine term and the absolute value are taken component-wise.

The cost function $f_{\mathrm{T}}(\boldsymbol{p})$ is to be minimized subject to the following inequality and equality constraints:
(a) *generator capacity constraints*

$$p_i^{\min} \leq p_i \leq p_i^{\max}, \quad i = 1, \ldots, n \tag{7.2}$$

where $p_i^{\min}$ and $p_i^{\max}$ are the lower and upper power generating limits of the ith unit (MW);
(b) *real power balance constraint*

$$\sum_{i=1}^{n} p_i = p_{\mathrm{D}} + p_{\mathrm{L}}(\boldsymbol{p}), \tag{7.3}$$

where p_{D} is the power demand (MW) and $p_{\mathrm{L}}(\boldsymbol{p})$ stands for the power loss (MW) expressed as

$$p_{\mathrm{L}}(\boldsymbol{p}) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_i B_{ij} p_j + \sum_{i=1}^{n} b_i^0 p_i + b^{00}, \tag{7.4}$$

or in matrix notation $p_{\mathrm{L}}(\boldsymbol{p}) = \boldsymbol{p}^\top \boldsymbol{B} \boldsymbol{p} + \boldsymbol{p}^\top \boldsymbol{b}^0 + b^{00}$. Coefficients B_{ij}, b_i^0, b^{00} are the transmission loss coefficients (B-coefficients) given by the elements of the square matrix $\boldsymbol{B}$ of size $n \times n$, the vector $\boldsymbol{b}^0$ of length n, and the constant b^{00}, respectively. The matrix $\boldsymbol{B}$ is symmetric positive-definite, hence $p_{\mathrm{L}}(\boldsymbol{p})$ is a convex quadratic function of $\boldsymbol{p}$.

To summarize, we consider the following optimization problem, that we will refer to as the ELDP:

$$\begin{aligned} &\min_{\boldsymbol{p} \in \mathbb{R}^n} f_{\mathrm{T}}(\boldsymbol{p}) \quad (7.1) \\ &\text{subject to (7.2) and (7.3).} \end{aligned} \tag{7.5}$$

7.2.2 Geometry of the Feasible Set

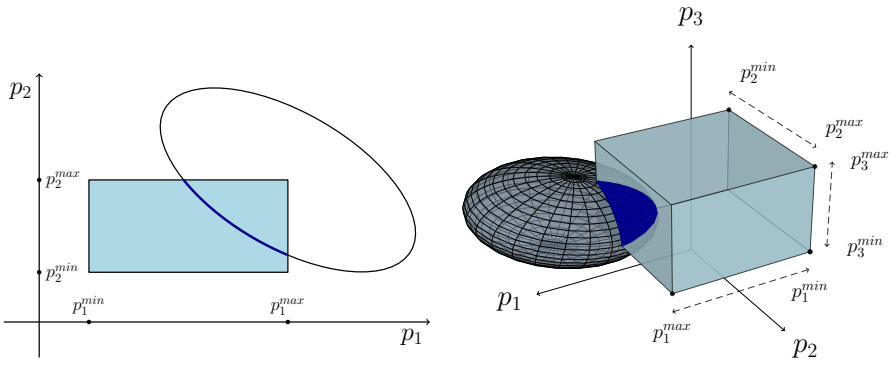
In this subsection, we explore the geometrical interpretation and consequences of constraints (7.2) and (7.3).

The bound constraints (7.2) force $\boldsymbol{p}$ to lie inside a box whose faces are parallel to the reference frame. The equality constraint (7.3) imposes that $\boldsymbol{p}$ lies on a quadric surface, specifically on an ellipsoid since $\boldsymbol{B}$ is positive definite. The feasible set of (7.5), denoted by Ω, is thus

composed of the points on the ellipsoid that are also inside the box:

$$\begin{aligned}\Omega &:= \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} \text{ satisfies (7.2) and (7.3)}\} \\ &= \left\{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p}^{\min} \leq \boldsymbol{p} \leq \boldsymbol{p}^{\max},\ \boldsymbol{p}^\top \boldsymbol{B}^{\text{ell}} \boldsymbol{p} + \boldsymbol{p}^\top \boldsymbol{b}^{\text{ell}} + c^{\text{ell}} = 0\right\},\end{aligned} \tag{7.6}$$

where $\mathbf{1}$ is the vector of all ones, $\boldsymbol{B}^{\text{ell}} = \boldsymbol{B}$, $\boldsymbol{b}^{\text{ell}} = \mathbf{b^0} - \mathbf{1}$, and $c^{\text{ell}} = b^{00} + p_d$. In general, Ω may be composed of more than one connected component, but we did not observe this in practical ELDP instances. The feasible set Ω is depicted in Figure 7.1 for the cases $n = 2$ and $n = 3$.



(a) $n = 2$ (b) $n = 3$

Figure 7.1: Representation of the $\mathbf{p}$-space along with the constraints (inside the box and on the ellipsoid) and the feasible set Ω (darker blue). Notice that Ω has measure zero in $\mathbb{R}^n$.

7.2.3 Structure of the Optimization Landscape

In this subsection, we provide some insight about the cost function $f_{\text{T}}(\boldsymbol{p})$ defined in (7.1). A first observation is that this function is separable, i.e., it consists in a sum of components f^i, each depending only on p_i, the ith component of $\boldsymbol{p}$. However, problem (7.5) does not decompose into independent one-dimensional problems because the components of $\boldsymbol{p}$ are coupled through the equality constraint (7.3). The cost function $f_{\text{T}}(\boldsymbol{p})$ is continuous but not everywhere differentiable. Indeed, at any point $\boldsymbol{q} \in \mathbb{R}^n$ for which one or more components q_i cancel the corresponding

sine term in f^i, the gradient of f_{T} does not exist because of the absolute value. These points, termed *nondifferentiable points* or *kink points*, are given by the following expression:

$$\boldsymbol{q} = (q_1, \ldots, q_n),\ \exists i \text{ s.t. } q_i = p_i^{\min} + \frac{k\pi}{e_i}, k \in \mathbb{N}. \tag{7.7}$$

Nevertheless, at all other points $\boldsymbol{p}$ in $\mathbb{R}^n$, the gradient of f_{T} can be computed:

$$\nabla f_{\mathrm{T}}(\boldsymbol{p}) = 2\mathrm{Diag}(\boldsymbol{a})\boldsymbol{p} + \boldsymbol{b} - \boldsymbol{d} \circ \boldsymbol{e} \circ \cos\boldsymbol{\theta} \circ \mathrm{sign}(\sin\boldsymbol{\theta}), \tag{7.8}$$

where $\boldsymbol{\theta}$ stands for $\left[\boldsymbol{e} \circ (\boldsymbol{p}^{\min} - \boldsymbol{p})\right]$. The function $f_{\mathrm{T}}(\boldsymbol{p})$ is thus piecewise smooth (as defined in [Roc03]) and one can use the canvas of Clarke's generalized calculus [Cla75] to compute its generalized gradient at the nondifferentiable points. This analysis will be carried out in Section 7.4.

Now, recall that problem (7.5) consists of minimizing $f_{\mathrm{T}}(\boldsymbol{p})$ over the feasible set Ω. As depicted in Figure 7.2 for an illustrative 2-dimensional case, the restriction of $f_{\mathrm{T}}(\boldsymbol{p})$ to Ω presents a partially smooth and multimodal landscape.

If the portion Ω of the ellipsoid is not too large, a similar representation can be obtained for a 3-dimensional case, by projecting the feasible set Ω onto the nearest 2-dimensional plane; see Figure 7.3, where the observations made for the 2-dimensional case can also be noted.

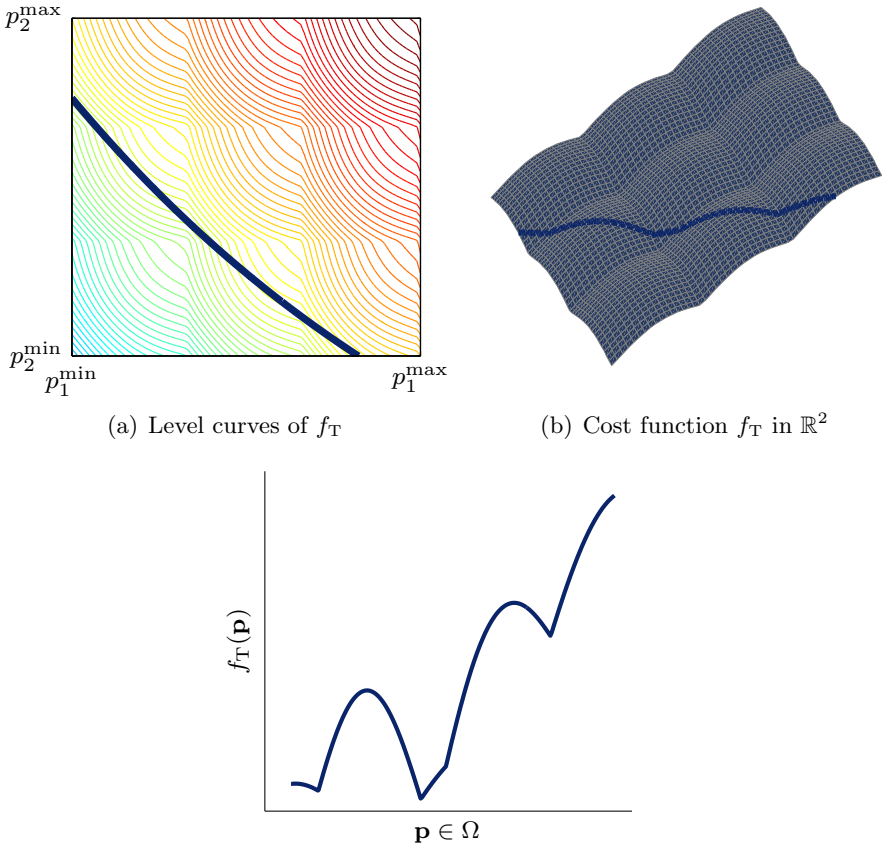
7.2.4 Summary of the Optimization Challenges

From the previous sections, it appears that the ELDP is a challenging optimization problem on three different levels: the geometry of its feasible set, the non-differentiability of its cost function, and the multimodal aspect of its landscape. The rest of this chapter presents our approach to deal with these difficulties.

7.3 Optimization Exploiting the Geometry of Ω

The proposed local method can be viewed as an adaptation of the classical line-search scheme

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^k + \alpha^k \boldsymbol{d}^k, \tag{7.9}$$



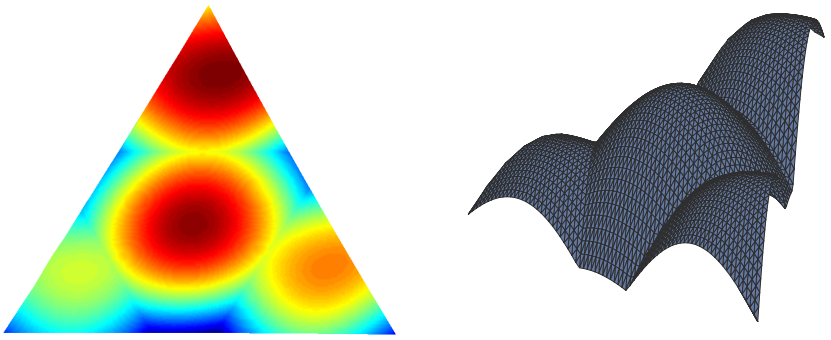
(a) Level curves of f_{T}

(b) Cost function f_{T} in $\mathbb{R}^2$

(c) Restriction to Ω

Figure 7.2: Landscape of the cost function $f_{\mathrm{T}}(\boldsymbol{p})$ for an illustrative 2-dimensional case. (a) Level curves of $f_{\mathrm{T}}(\boldsymbol{p})$ and the ellipsoid constraint. (b) Graph of the cost function $f_{\mathrm{T}}(\boldsymbol{p})$. (c) Graphs of the restriction of $f_{\mathrm{T}}(\boldsymbol{p})$ to Ω. The cost function is multimodal and piecewise smooth.

where $\boldsymbol{p}^k$ and $\boldsymbol{p}^{k+1}$ denote the current and next iterates, $\boldsymbol{d}^k$ is the search direction, and α^k is the step length. In this section, with property (i)—feasibility—in mind, we generalize the "+" operation by means of the concept of retraction and we introduce "admissibility" conditions on $\boldsymbol{d}^k$ in order to guarantee that $\boldsymbol{p}^{k+1}$ remains in Ω for all α^k sufficiently small. Then, in Section 7.4, we will show how to choose $\boldsymbol{d}^k$ as the



(a) Level curves (b) Restriction of f_{T} to $\hat{\Omega}$

Figure 7.3: Landscape of the cost function $f_{\mathrm{T}}(\boldsymbol{p})$ for an illustrative 3-dimensional case. This representation is made possible by projecting the 3 dimensional feasible set Ω onto the nearest plane. (a) Level curves of the restriction of $f_{\mathrm{T}}(\boldsymbol{p})$ to the projection $\hat{\Omega}$ of Ω. (b) Graph of the restriction of $f_{\mathrm{T}}(\boldsymbol{p})$ to $\hat{\Omega}$. The cost function is multimodal and piecewise smooth.

steepest admissible direction. Finally, the selection of α^k, as well as other implementation details, will be addressed in Section 7.5.

7.3.1 The Ellipsoid Manifold

As explained in Section 7.2.2, the feasible set Ω of the ELDP (7.5) has a rich geometrical structure that we would like to exploit. The set of points that satisfy the equality constraint (7.3) is an ellipsoid centred at

$$\boldsymbol{a}^{\mathrm{ell}} = -\frac{1}{2}(\boldsymbol{B}^{\mathrm{ell}})^{-1}\boldsymbol{b}^{\mathrm{ell}}. \tag{7.10}$$

This smooth surface has a natural structure of a manifold, specifically of an $(n-1)$-dimensional submanifold of $\mathbb{R}^n$. We will call it the ellipsoid manifold:

$$\begin{aligned}\mathcal{E}^{n-1} :=& \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} \text{ satisfies (7.3)}\} \\ =& \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p}^\top \boldsymbol{B}^{\mathrm{ell}} \boldsymbol{p} + \boldsymbol{p}^\top \boldsymbol{b}^{\mathrm{ell}} + c^{\mathrm{ell}} = 0\}.\end{aligned} \tag{7.11}$$

The gist of the proposed Riemannian optimization approach is to restrict the optimization domain $\mathbb{R}^n$ to the constraint manifold, in this

case with the resulting advantage that every iterate belongs to $\mathcal{E}^{n-1}$, i.e., satisfies the equality constraint (7.3). The ELDP (7.5) then becomes:

$$\min_{\boldsymbol{p}\in\mathcal{E}^{n-1}} f_{\mathrm{T}}(\boldsymbol{p}) \quad \text{subject to} \quad \boldsymbol{p}^{\min} \leq \boldsymbol{p} \leq \boldsymbol{p}^{\max}.$$

We mention that an alternative way, used by several ELDP heuristics, of respecting the equality constraint (7.3) is to resort to a slack variable, which amounts to performing the optimization on $(n-1)$ variables while computing the last variable explicitly. A difficulty with this approach is that the description of the cost function f_{T}, of the feasible set Ω, and of the nondifferentiable points in terms of the $(n-1)$ remaining variables becomes more intricate. This difficulty is avoided in the Riemannian approach.

7.3.2 Riemannian Optimization Ingredients

The classical line-search scheme (7.9) is not meant to produce iterates that remain on a submanifold such as $\mathcal{E}^{n-1}$. In this subsection, we provide the needed ingredients of differential geometry (the branch of mathematics that studies manifolds) in order to produce iterates on an abstract submanifold $\mathcal{M}$ of $\mathbb{R}^n$. Then, in Section 7.3.3, we will specialize the ingredients to the manifold $\mathcal{E}^{n-1}$, and finally, in Section 7.3.5, we will reintroduce the bound constraints (7.2).

Instrumental to the definition of an admissible direction $\boldsymbol{d}^k$ is the notion of *tangent space* to $\mathcal{M}$ at a point $\boldsymbol{p} \in \mathcal{M}$, denoted by $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, and defined as the following vector subspace of $\mathbb{R}^n$:

$$\mathrm{T}_{\boldsymbol{p}}\mathcal{M} = \{\boldsymbol{\xi} \in \mathbb{R}^n : \exists c : \mathbb{R} \to \mathcal{M} \text{ with } c(0) = \boldsymbol{p},\ c'(0) = \boldsymbol{\xi}\},$$

where $c'(0)$ is the usual derivative at 0 of the curve c (assumed to exist). From this definition, we see that $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is the set of vectors that are tangent to the manifold at $\boldsymbol{p}$. Geometrically, this notion coincides with the concept of tangent plane to a smooth surface, as depicted in Figure 7.4. The tangent bundle $\mathrm{T}\mathcal{M}$ is the collection of the tangent spaces at all $\boldsymbol{p} \in \mathcal{M}$.

The notion of *steepest* admissible direction will require a norm on $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$. Each tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is a vector space and as such can be endowed with an inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{p}}$. The natural way of doing this is by restricting the canonical inner product of $\mathbb{R}^n$ to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$, i.e.,

$$\langle ., . \rangle_{\boldsymbol{p}} : \mathrm{T}_{\boldsymbol{p}}\mathcal{M} \times \mathrm{T}_{\boldsymbol{p}}\mathcal{M} \to \mathbb{R} : (\boldsymbol{\xi}, \boldsymbol{\zeta}) \mapsto \langle \boldsymbol{\xi}, \boldsymbol{\zeta} \rangle_{\boldsymbol{p}} = \langle \boldsymbol{\xi}, \boldsymbol{\zeta} \rangle = \boldsymbol{\xi}^{\top}\boldsymbol{\zeta}.$$

We then say that $\mathcal{M}$ is a *Riemannian submanifold* of $\mathbb{R}^n$. The inner product induces a notion of norm: $\|\boldsymbol{\xi}\|_{\boldsymbol{p}} = \langle \boldsymbol{\xi}, \boldsymbol{\xi} \rangle_{\boldsymbol{p}}^{1/2}$.

The *normal space* to $\mathcal{M}$ at a point $\boldsymbol{p} \in \mathcal{M}$, denoted as $\mathrm{N}_{\boldsymbol{p}}\mathcal{M}$, is the orthogonal complement of $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ in $\mathbb{R}^n$:

$$\mathrm{N}_{\boldsymbol{p}}\mathcal{M} = (\mathrm{T}_{\boldsymbol{p}}\mathcal{M})^{\perp}.$$

One can then compute the *projection* $P_{\boldsymbol{p}}(\boldsymbol{v})$ of a vector $\boldsymbol{v} \in \mathbb{R}^n$ onto $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ by removing the normal component of $\boldsymbol{v}$.

We now turn to the generalization of the "+" operation of (7.9). A *retraction* on $\mathcal{M}$ [ADM+02, AMS08] is a smooth mapping R from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ that satisfies $R(0_{\boldsymbol{p}}) = \boldsymbol{p}$ for all $\boldsymbol{p}$ (where $0_{\boldsymbol{p}}$ denotes the origin of $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$) and $\frac{\mathrm{d}}{\mathrm{d}t} R(t\boldsymbol{\xi}_{\boldsymbol{p}})\big|_{t=0} = \boldsymbol{\xi}_{\boldsymbol{p}}$ for all $\boldsymbol{\xi}_{\boldsymbol{p}} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{M}$. The restriction of R to $\mathrm{T}_{\boldsymbol{p}}\mathcal{M}$ is denoted by $R_{\boldsymbol{p}}$. Observe that R given by $R_{\boldsymbol{p}}(\alpha \boldsymbol{d}) = \boldsymbol{p} + \alpha \boldsymbol{d}$ is a valid retraction in $\mathbb{R}^n$, hence the generalization. For the ellipsoid manifold $\mathcal{E}^{n-1}$, a possible retraction is given by (7.15).

Using these tools, one can adapt the iterative process (7.9) as follows: assuming that $\boldsymbol{p}^k \in \mathcal{M}$, construct $\boldsymbol{p}^{k+1} \in \mathcal{M}$ according to

$$\boldsymbol{p}^{k+1} = R_{\boldsymbol{p}^k}\left(\alpha^k P_{\boldsymbol{p}}^k(\boldsymbol{d}^k)\right),\ \alpha^k \in \mathbb{R},\ \boldsymbol{d}^k \in \mathbb{R}^n.$$

7.3.3 Optimization Ingredients on $\mathcal{E}^{n-1}$

We now specifically provide the aforementioned tools for the ellipsoid manifold $\mathcal{E}^{n-1}$. A schematic diagram is given in Figure 7.4.

Let $c(t)$ be a curve on $\mathcal{E}^{n-1}$ parametrized by $t \in \mathbb{R}$:

$$c : \mathbb{R} \to \mathcal{E}^{n-1} : t \to c(t),\ \text{s.t. } c(0) = \boldsymbol{p},\ \dot{c}(0) = \boldsymbol{\xi}.$$

For all $t \in \mathbb{R}$, the ellipsoid equation gives $c(t)^\top \boldsymbol{B}^{\mathrm{ell}} c(t) + c(t)^\top \boldsymbol{b}^{\mathrm{ell}} + c^{\mathrm{ell}} = 0$. Taking the derivative with respect to t and evaluating at $t = 0$ (substituting for $\boldsymbol{p}$ and $\boldsymbol{\xi}$), one obtains:

$$\boldsymbol{\xi}^\top \boldsymbol{B}^{\mathrm{ell}} \boldsymbol{p} + \boldsymbol{p}^\top \boldsymbol{B}^{\mathrm{ell}} \boldsymbol{\xi} + \boldsymbol{\xi}^\top \boldsymbol{b}^{\mathrm{ell}} = 0.$$

Since $\boldsymbol{B}^{\mathrm{ell}}$ is a symmetric matrix, one can rewrite:

$$\boldsymbol{\xi}^T (2\boldsymbol{B}^{\mathrm{ell}} \boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}) = 0.$$

The *tangent space* is thus defined as follows:

$$\mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1} = \{\boldsymbol{\xi} \in \mathbb{R}^n : \boldsymbol{\xi}^\top (2\boldsymbol{B}^{\mathrm{ell}} \boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}) = 0\}. \tag{7.12}$$

The *normal space* is then obtained by:

$$\begin{aligned} \mathrm{N}_{\boldsymbol{p}}\mathcal{E}^{n-1} &= \{\boldsymbol{\nu} \in \mathbb{R}^n : \boldsymbol{\xi}^T \boldsymbol{\nu} = 0,\ \forall \boldsymbol{\xi} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1}\} \\ &= \{\tau\,(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}),\ \tau \in \mathbb{R}\}. \end{aligned} \tag{7.13}$$

The *projection* $P_{\boldsymbol{p}}(\boldsymbol{v})$ of a vector $\boldsymbol{v} \in \mathbb{R}^n$ onto $T_{\boldsymbol{p}}\mathcal{E}^{n-1}$ can then be constructed so as to remove the normal component of $\boldsymbol{v}$:

$$P_{\boldsymbol{p}}(v) = \boldsymbol{v} - \tau(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}),$$

where the value of $\tau \in \mathbb{R}$ must be determined to ensure that $P_{\boldsymbol{p}}(\boldsymbol{v})$ belongs to $\mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1}$:

$$(\boldsymbol{v} - \tau(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}))^{\top}(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}}) = 0,$$

which yields

$$\tau = \frac{\boldsymbol{v}^{\top}(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}})}{\|(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}})\|^2}.$$

Defining $\boldsymbol{n}_{\boldsymbol{p}} = \frac{(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}})}{\|(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{p} + \boldsymbol{b}^{\mathrm{ell}})\|}$ as the unit normal vector to the ellipsoid at $\boldsymbol{p}$, the projection becomes:

$$P_{\boldsymbol{p}}(\boldsymbol{v}) = \boldsymbol{v} - \boldsymbol{n}_{\boldsymbol{p}}\boldsymbol{n}_{\boldsymbol{p}}^{\top}\boldsymbol{v} = (I - \boldsymbol{n}_{\boldsymbol{p}}\boldsymbol{n}_{\boldsymbol{p}}^{\top})\boldsymbol{v}. \tag{7.14}$$

Finally, the *retraction* at a point $\boldsymbol{p} \in \mathcal{E}^{n-1}$ of a tangent vector $\boldsymbol{\xi} \in T_{\boldsymbol{p}}\mathcal{E}^{n-1}$ can be defined, using the following scaling:

$$\begin{aligned} R_{\boldsymbol{p}}(\boldsymbol{\xi}) &= \boldsymbol{a}^{\mathrm{ell}} + \beta\left(\boldsymbol{p} + \boldsymbol{\xi} - \boldsymbol{a}^{\mathrm{ell}}\right) \\ &= \boldsymbol{a}^{\mathrm{ell}} + \beta\boldsymbol{w} = \boldsymbol{q}, \end{aligned} \tag{7.15}$$

where $\boldsymbol{a}^{\mathrm{ell}}$ is the center of the ellipsoid given by (7.10) and $\beta \in \mathbb{R}$ must be chosen so as to satisfy $\boldsymbol{q} \in \mathcal{E}^{n-1}$:

$$\boldsymbol{q}^{\top}\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{q} + \boldsymbol{q}^{\top}\boldsymbol{b}^{\mathrm{ell}} + c^{\mathrm{ell}} = 0, \tag{7.16}$$

which yields:

$$\begin{aligned} \beta^2\left(\boldsymbol{w}^{\top}\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{w}\right) + \beta\left(\boldsymbol{w}^{\top}(2\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{a}^{\mathrm{ell}} + \boldsymbol{b}^{\mathrm{ell}})\right) \\ + \left(\boldsymbol{a}^{\mathrm{ell}\top}(\boldsymbol{B}^{\mathrm{ell}}\boldsymbol{a}^{\mathrm{ell}} + \boldsymbol{b}^{\mathrm{ell}}) + c^{\mathrm{ell}}\right) = 0 \end{aligned} \tag{7.17}$$

Equation (7.17) defines a parabola in β. The roots of this parabola correspond to the two intersections with the ellipsoid of the line going trough $\boldsymbol{a}^{\text{ell}}$ and $(\boldsymbol{p}+\boldsymbol{\xi})$ (see $\boldsymbol{q}$ and $\boldsymbol{q'}$ in Figure 7.4). Among these two points, the nearest to $(\boldsymbol{p}+\boldsymbol{\xi})$ corresponds to the closest root to 1, which is therefore chosen for β. The fact that $R_{\boldsymbol{p}}$ is a retraction follows from [AM12, theorem 15].

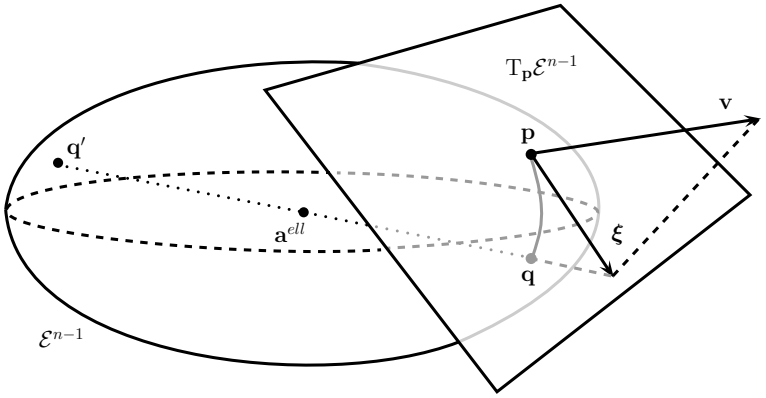


Figure 7.4: Illustration of the optimization tools on the ellipsoid manifold $\mathcal{E}^{n-1}$. The vector $\boldsymbol{v} \in \mathbb{R}^n$ is projected onto the tangent space $\mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1}$: $\boldsymbol{\xi} = P_{\boldsymbol{p}}(\boldsymbol{v})$. The retraction of the tangent vector $\boldsymbol{\xi}$ is then computed: $\boldsymbol{q} = R_{\boldsymbol{p}}(\boldsymbol{\xi})$.

7.3.4 Restriction to Sub-Ellipsoids

As will be discussed in Section 7.4.3, it will prove useful at times to consider the restriction of the ellipsoid $\mathcal{E}^{n-1}$ obtained by fixing some coordinates in $\boldsymbol{p} = (p_1, \ldots, p_n)^\top$. Let $\mathcal{C} \subset \{1, \ldots, n\}$ be the indices of these constant coordinates with $|\mathcal{C}| = n_c < n$, and let $\boldsymbol{P} = (P_1, \ldots, P_{n_c})^\top$ be the corresponding constants. The following notation is introduced: given a matrix $\boldsymbol{M}$, a vector $\boldsymbol{v}$ and two sets of indices $\mathcal{I}_1$ and $\mathcal{I}_2$, the sub-matrix $[M_{i,j}]_{i\in\mathcal{I}_1, j\in\mathcal{I}_2}$ is denoted as $\boldsymbol{M}_{\mathcal{I}_1,\mathcal{I}_2}$ and the sub-vector $[v_i]_{i\in\mathcal{I}_1}$ is denoted as $\boldsymbol{v}_{\mathcal{I}_1}$.

We are interested in the following set:

$$\mathcal{E}^{n-1}_{\mathcal{C},\boldsymbol{P}} := \{\boldsymbol{p} \in \mathcal{E}^{n-1} \ : \ \boldsymbol{p}_{\mathcal{C}} = \boldsymbol{P}\}.$$

This set describes a Riemannian manifold which is the intersection of the ellipsoid $\mathcal{E}^{n-1}$ with the intersection of the n_c axis-aligned hyperplanes defined by $\boldsymbol{p}_{\mathcal{C}} = \boldsymbol{P}$. This manifold is in fact again an ellipsoid of dimension $(n - n_c - 1)$, as depicted in Figure 7.5.

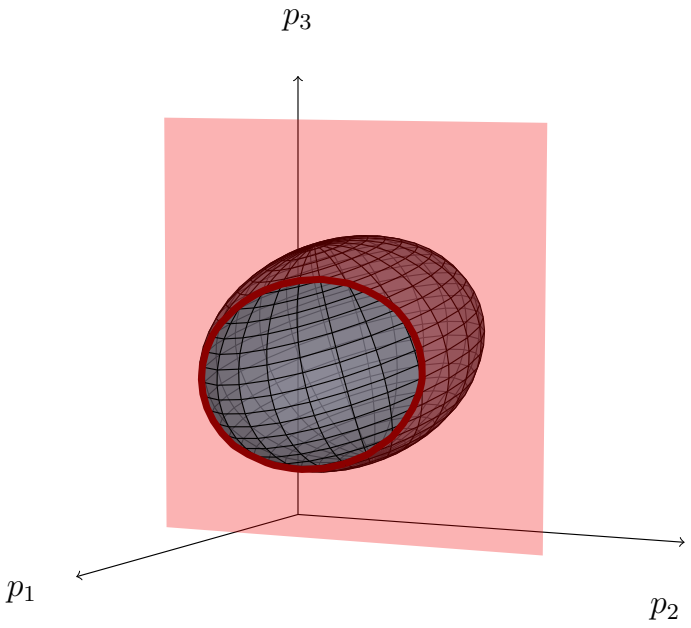


Figure 7.5: Illustration of a sub-ellipsoid for $n = 3$, with $\mathcal{C} = \{1\}$ and $\mathcal{F} = \{2, 3\}$.

Letting $\mathcal{F} = \{1, \dots, n\} \setminus \mathcal{C}$ be the set of free coordinates, this sub-ellipsoid is given by the following equation:

$$\mathcal{E}^{n-1}_{\mathcal{C},\boldsymbol{P}} := \left\{ \boldsymbol{p} \in \mathbb{R}^n \ : \ \boldsymbol{p}_{\mathcal{F}}^\top \widehat{\boldsymbol{B}^{\text{ell}}} \boldsymbol{p}_{\mathcal{F}} + \boldsymbol{p}_{\mathcal{F}}^\top \widehat{\boldsymbol{b}^{\text{ell}}} + \widehat{c^{\text{ell}}} = 0, \ \boldsymbol{p}_{\mathcal{C}} = \boldsymbol{P} \right\}, \quad (7.18)$$

where

$$\begin{aligned} \widehat{\boldsymbol{B}^{\text{ell}}} &= \boldsymbol{B}^{\text{ell}}_{\mathcal{F},\mathcal{F}}, \\ \widehat{\boldsymbol{b}^{\text{ell}}} &= \left(\boldsymbol{b}^{\text{ell}}_{\mathcal{F}} + 2\boldsymbol{B}^{\text{ell}}_{\mathcal{F},\mathcal{C}} \boldsymbol{p}_{\mathcal{C}} \right), \\ \widehat{c^{\text{ell}}} &= \boldsymbol{p}_{\mathcal{C}}^\top \left(\boldsymbol{B}^{\text{ell}}_{\mathcal{C},\mathcal{C}} \boldsymbol{p}_{\mathcal{C}} + \boldsymbol{b}^{\text{ell}}_{\mathcal{C}} \right) + c^{\text{ell}}. \end{aligned}$$

The center of this new ellipsoid is then given by $\widehat{a}^{\text{ell}}_{\mathcal{F}} = -\frac{1}{2}(\widehat{\boldsymbol{B}^{\text{ell}}})^{-1}\widehat{\boldsymbol{b}^{\text{ell}}}$ and $\widehat{a}_{\mathcal{C}} = \boldsymbol{P}$.

The ingredients presented in the previous section can now be adapted to the sub-ellipsoid when needed. In particular, when computing the retraction $\boldsymbol{q} = R_{\boldsymbol{p}}(\boldsymbol{\xi})$ of a vector $\boldsymbol{\xi} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1}_{\mathcal{C},\boldsymbol{p}_\mathcal{C}}$, one expects $\boldsymbol{q}$ to remain on $\mathcal{E}^{n-1}_{\mathcal{C},\boldsymbol{p}_\mathcal{C}}$. With the definition (7.15) of R, this is not the case since the scaling factor β affects all of the coordinates of $\boldsymbol{q}$. However, the following retraction $\widehat{R}$ (where $\mathcal{C}$ is omitted in the notation as it can be deduced from the argument $\boldsymbol{\xi}$, see (7.44) for the robustified version) can be used to ensure the desired property:

$$\widehat{R}_{\boldsymbol{p}}(\boldsymbol{\xi}) = \boldsymbol{q}, \text{ with } \begin{cases} \boldsymbol{q}_\mathcal{C} = \boldsymbol{p}_\mathcal{C}, \\ \boldsymbol{q}_\mathcal{F} = \hat{\boldsymbol{a}} + \beta\left(\boldsymbol{p}_\mathcal{F} + \boldsymbol{\xi}_\mathcal{F} - \hat{\boldsymbol{a}}\right), \end{cases} \tag{7.19}$$

where β is computed using (7.17) with the corresponding $\widehat{\boldsymbol{B}^{\text{ell}}}$, $\widehat{\boldsymbol{b}^{\text{ell}}}$, $\widehat{c^{\text{ell}}}$ and $\widehat{\boldsymbol{a}^{\text{ell}}}$. The difference between the two retractions R and $\hat{R}$ is illustrated in Figure 7.6.

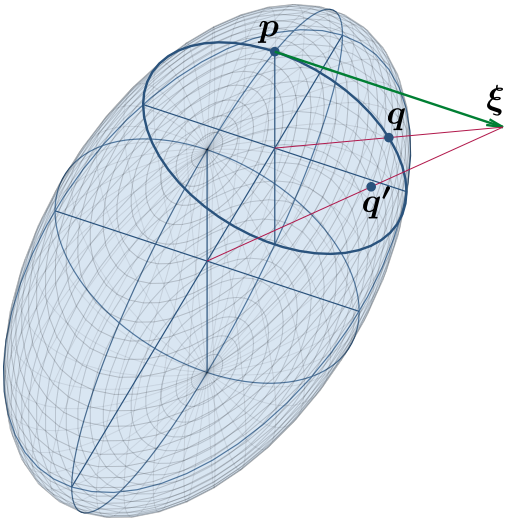


Figure 7.6: Retraction of the tangent vector $\boldsymbol{\xi}$ at the point $\boldsymbol{p}$. When using the retraction R given by (7.15), the point $\boldsymbol{q'}$ is obtained, outside of the subellipsoid going through $\boldsymbol{p}$. When using the retraction $\hat{R}$ (7.19) relative to the center of this subellipsoid, the point $\boldsymbol{q}$ is obtained, belonging to the subellipsoid.

Noting that $\hat{R}_{\boldsymbol{p}}(\boldsymbol{\xi}) = R_{\boldsymbol{p}}(\boldsymbol{\xi})$ when $\mathcal{C} = \emptyset$, equation (7.19) will be used whenever a retraction is needed in the remainder of this chapter.

7.3.5 Respecting the Bound Constraints

So far in this section, we have considered performing optimization on $\mathcal{E}^{n-1}$. However, as presented in Section 7.2.2, the feasible set Ω is only a portion of $\mathcal{E}^{n-1}$, delimited by the bound constraints (7.2). In view of the sought property (i) mentioned in the introduction, we want to produce iterates that remain within these bounds at all times. To this end, a possible approach would be to penalize the cost function f_{T} with a log barrier term:

$$\widehat{f_{\mathrm{T}}}(\boldsymbol{p}) = \begin{cases} f_{\mathrm{T}}(\boldsymbol{p}) + \mu_k \phi(\boldsymbol{p}) & \text{if } \boldsymbol{p}^{\min} \leq \boldsymbol{p} \leq \boldsymbol{p}^{\max}, \\ +\infty & \text{otherwise,} \end{cases}$$
$$\text{with } \phi(\boldsymbol{p}) = \ln(\boldsymbol{p} - \boldsymbol{p}^{\min}) + \ln(\boldsymbol{p}^{\max} - \boldsymbol{p}),$$

where $\mu_k \downarrow 0$ when $k \to \infty$. This technique aims at repelling iterates from the boundaries of Ω to avoid premature convergence to nonstationary points. Other techniques to handle the bound constrains include penalty terms and augmented Lagrangian (see, e.g., chapter 17 in [NW06]), but they do not ensure feasibility of the iterates.

We choose to adopt an alternative to these approaches. The principle consists of computing the steepest admissible direction $\boldsymbol{d}^k$ at the current iterate $\boldsymbol{p}^k$, and then setting $\boldsymbol{p}^{k+1} = \widehat{R}_{\boldsymbol{p}}(\alpha^k \boldsymbol{d}^k)$ where α^k is selected by means of a line-search technique. In this context, by admissible direction, we mean the following.

Definition 7.1. An *admissible direction* at $\boldsymbol{p} \in \Omega$ is a vector $\boldsymbol{d} \in \mathrm{T}_{\boldsymbol{p}}\mathcal{E}^{n-1}$ for which there exists $\epsilon > 0$ such that $\widehat{R}_{\boldsymbol{p}}(\alpha \boldsymbol{d})$ belongs to Ω for all $\alpha \in [0, \epsilon]$.

The computation of the steepest admissible direction will be presented in Section 7.4 and the line-search technique will be discussed in Section 7.5.3

We already mention that, in order to be functional in combination with the chosen basic Armijo-type line-search technique, the proposed approach requires a robustification mechanism, described in Section 7.5.2, both for the capacity constraints (7.2) and for the nondifferentiable points (7.7). Otherwise, since the nondifferentiable points and boundary points form a zero-measure set, all the iterates would be likely to be considered differentiable points that strictly satisfy the bound constraints (7.2), with the consequence that the computed steepest admissible direction would merely be the projected gradient (7.20)

at all times, yielding unsatisfactory convergence as illustrated by the squares in Figure 7.9.

7.4 Subgradient Descent for the ELDP

This section chiefly concerns the computation of the steepest admissible direction for the ELDP (7.5). We will proceed gradually. The concept of subgradient is first recalled in an unconstrained setting in Section 7.4.1, then it is generalized to the Riemannian setting in Section 7.4.2 with a view towards considering the ELDP cost function on the ellipsoid manifold $\mathcal{E}^{n-1}$, which is done in Section 7.4.3, before reintroducing the bound constraints in Section 7.4.4.

As stated in Section 7.2.3, the cost function $f_{\mathrm{T}}(\boldsymbol{p})$ is piecewise smooth, i.e., differentiable almost everywhere. Let S, resp. D, denote the set of nondifferentiable points, resp. differentiable points, of f_{T} in Ω:

$$S = \left\{\boldsymbol{p} \in \Omega \ : \ \exists i \text{ s.t. } p_i = p_i^{min} + \frac{k\pi}{e_i}, k \in \mathbb{N}\right\}, \quad D = \Omega \setminus S.$$

We will also consider the set S^o, resp. D^o, of points of S, resp. D, that are not on the boundary of the box (7.2). Note that in ELDP instances, one must expect that both S^o and $S \setminus S^o$ are nonempty, and likewise for D^o and $D \setminus D^o$.

For $\boldsymbol{p} \in D^o$, the steepest admissible direction $\mathrm{grad} f_{\mathrm{T}}(\boldsymbol{p})$ is simply the projection of the gradient of $f_{\mathrm{T}}(\boldsymbol{p})$ onto $T_{\boldsymbol{p}}\mathcal{E}^{n-1}$:

$$\mathrm{grad} f_{\mathrm{T}}(\boldsymbol{p}) = P_{\boldsymbol{p}}\left(\nabla f_{\mathrm{T}}(\boldsymbol{p})\right), \tag{7.20}$$

where $P_{\boldsymbol{p}}$ is given by (7.14) and $\nabla f_{\mathrm{T}}(\boldsymbol{p})$ by (7.8). For all other points $\boldsymbol{q} \in \Omega \setminus D^o$, describing the steepest admissible direction requires more sophisticated developments that we present in the rest of this section.

7.4.1 Subgradient Descent

Given a nonsmooth, nonconvex function $f : \mathbb{R}^n \to \mathbb{R}$, Clarke's generalized directional derivative is given by:

$$f^\circ(\boldsymbol{x}; \boldsymbol{d}) = \limsup_{\boldsymbol{y} \to \boldsymbol{x},\, t \downarrow 0} \frac{f(\boldsymbol{y} + t\boldsymbol{d}) - f(\boldsymbol{y})}{t}.$$

The generalized gradient of f, noted $\partial f(\boldsymbol{x})$, is then defined as the set of the subgradients $\boldsymbol{s}$ of $f^\circ(\boldsymbol{x};\cdot)$:

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{s} \in \mathbb{R}^n : f^\circ(\boldsymbol{x};\boldsymbol{v}) \geq \boldsymbol{v}^\top \boldsymbol{s}, \forall \boldsymbol{v} \in \mathbb{R}^n \right\}.$$

For a general nonsmooth function, this set can be difficult to describe in practice. However, if the function is locally Lipschitz, which is the case of the ELDP cost function (7.1), then the generalized gradient at $\boldsymbol{x}$ is the convex hull of all points $\boldsymbol{s}$ of the form

$$\boldsymbol{s} = \lim_{i \to \infty} \nabla f(\boldsymbol{x}_i),$$

where $\{\boldsymbol{x}_i\}$ is a sequence converging to $\boldsymbol{x}$ such that f is differentiable at each $\boldsymbol{x}_i$ (see [Cla76, proposition 5]). If moreover, as is the case of the ELDP cost function (7.1), $f : \mathbb{R}^n \to \mathbb{R}$ can be expressed as the pointwise maximum of m smooth functions $f_j : \mathbb{R}^n \to \mathbb{R}$, i.e.,

$$f(\boldsymbol{x}) = \max_{j=1,\ldots,m} f_j(\boldsymbol{x}), \tag{7.21}$$

then the generalized gradient can be simply described as

$$\partial f(\boldsymbol{x}) = \operatorname{co}\left\{ \nabla f_j(\boldsymbol{x}) \mid j \in \mathcal{I}_f(\boldsymbol{x}) \right\}, \tag{7.22}$$

where $\operatorname{co}\{\cdot\}$ denotes the convex hull and $\mathcal{I}_f(\boldsymbol{x})$ denotes the set of indices for which the maximum in (7.21) is attained, i.e.,

$$\mathcal{I}_f(\boldsymbol{x}) = \{ j \in \{1,\ldots,m\} \mid f(\boldsymbol{x}) = f_j(\boldsymbol{x}) \}. \tag{7.23}$$

Using this framework, and without considering any constraint for the time being, the steepest descent direction $\boldsymbol{d}^k$ is given by the opposite of the shortest vector in $\partial f(\boldsymbol{x}^k)$ [BLO05, lemma 2.1], which can be obtained by solving the following quadratic programming problem:

$$\min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| \sum_{j \in \mathcal{I}_f(\boldsymbol{x}^k)} \lambda_j \nabla f_j(\boldsymbol{x}^k) \right\|^2. \tag{7.24}$$

In practice, when the set $\partial f(\boldsymbol{x}^k)$ is composed of a single element $\nabla f_j(\boldsymbol{x}^k)$, the subproblem becomes trivial as the descent direction is simply given by $\boldsymbol{d}^k = \nabla f_j(\boldsymbol{x}^k)$. When $\partial f(\boldsymbol{x}^k)$ contains at least two elements, solving the subproblem allows one to compute a descent direction even though the function is not differentiable.

7.4.2 Riemannian subgradient descent

In order to extend the subgradient descent to a Riemannian manifold setting, one has to redefine the needed ingredients. Let $\mathcal{M}$ be a submanifold of $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ a piecewise smooth function, defined as the maximum of m smooth functions f_j. Using the tools presented in Section 7.3.2, the projected gradient of each function f_j at a point $\boldsymbol{x} \in \mathcal{M}$ is defined as follows:

$$\operatorname{grad} f_j(\boldsymbol{x}) = P_{\boldsymbol{x}}(\nabla f_j(\boldsymbol{x})) \quad \in T_{\boldsymbol{x}}\mathcal{M}.$$

The generalized projected gradient is then given by:

$$\begin{aligned} \operatorname{grad} f(\boldsymbol{x}) &= \operatorname{co}\left\{\operatorname{grad} f_j(\boldsymbol{x}) \mid j \in \mathcal{I}_f(\boldsymbol{x})\right\} \\ &= \operatorname{co}\left\{P_{\boldsymbol{x}}(\nabla f_j(\boldsymbol{x})) \mid j \in \mathcal{I}_f(\boldsymbol{x})\right\}. \end{aligned} \tag{7.25}$$

The steepest admissible direction $\boldsymbol{d}^k$ is obtained by computing the shortest vector in $\operatorname{grad} f(\boldsymbol{x}^k)$, solving the following quadratic program:

$$\begin{aligned} &\min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| \sum_{j \in \mathcal{I}(\boldsymbol{x}^k)} \lambda_j \operatorname{grad} f_j(\boldsymbol{x}^k) \right\|^2 \\ &= \min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| \sum_{j \in \mathcal{I}(\boldsymbol{x}^k)} \lambda_j P_{\boldsymbol{x}^k}(\nabla f_j(\boldsymbol{x}^k)) \right\|^2 \\ &= \min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| P_{\boldsymbol{x}^k} \left(\sum_{j \in \mathcal{I}_f(\boldsymbol{x}^k)} \lambda_j \nabla f_j(\boldsymbol{x}^k) \right) \right\|^2. \end{aligned} \tag{7.26}$$

The steepest-descent admissible direction is computed as follows:

$$\boldsymbol{d}^k = -P_{\boldsymbol{x}^k} \left(\sum_{j \in \mathcal{I}_f(\boldsymbol{x}^k)} \lambda_j^* \nabla f_j(\boldsymbol{x}^k) \right),$$

where $\boldsymbol{\lambda}^*$ is the solution of (7.26).

7.4.3 Application to the ELDP cost function on $\mathcal{E}^{n-1}$

In this subsection, we adapt the subgradient descent scheme to the ELDP, while temporarily ignoring the bound constraints (7.2) until Section 7.4.4. We first show that the ELDP cost function f_{T} (7.1) can be

expressed as the pointwise maximum of smooth functions, then we compute the associated generalized gradient and we formulate the search for a descent direction as a quadratic programming problem.

Recalling that the ELDP cost function $f_{\mathrm{T}}(\boldsymbol{p})$ is the sum of n functions $f^i(p_i)$ $(i = 1, \dots, n)$, let $\hat{f}^i(\boldsymbol{p}) \doteq f^i(p_i)$. Each of these functions can be decomposed into two components, $\hat{f}^i_{\mathrm{q}}(\boldsymbol{p}) = a_i p_i^2 + b_i p_i + c_i$ (the quadratic term) and $\hat{f}^i_{\mathrm{s}}(\boldsymbol{p}) = d_i \sin\left[e_i\left(p_i^{\min} - p_i\right)\right]$ (the sinusoidal term). The ELDP cost function can then be written as follows:

$$\begin{aligned} f_{\mathrm{T}}(\boldsymbol{p}) &= \sum_{i=1}^{n} \hat{f}^i_{\mathrm{q}}(\boldsymbol{p}) + \left|\hat{f}^i_{\mathrm{s}}(\boldsymbol{p})\right| \\ &= \sum_{i=1}^{n} \max\left(\hat{f}^i_{\mathrm{q}}(\boldsymbol{p}) - \hat{f}^i_{\mathrm{s}}(\boldsymbol{p}),\ \hat{f}^i_{\mathrm{q}}(\boldsymbol{p}) + \hat{f}^i_{\mathrm{s}}(\boldsymbol{p})\right). \end{aligned} \tag{7.27}$$

Clearly, f_{T} can be expressed as the maximum of 2^n functions $f_{\mathrm{T},j}$, taking all the possible combinations of the maximum arguments in (7.27):

$$\begin{aligned} f_{\mathrm{T}}(\boldsymbol{p}) &= \max_{j=1,\dots,2^n} f_{\mathrm{T},j}(\boldsymbol{p}) \\ &= \max_{j=1,\dots,2^n} \left(\sum_{i=1}^{n} \hat{f}^i_{\mathrm{q}}(\boldsymbol{p}) + \sum_{i=1}^{n} (-1)^{\left\lfloor \frac{j\%2^i}{2^{i-1}} \right\rfloor} \hat{f}^i_{\mathrm{s}}(\boldsymbol{p}) \right) \\ &= f_{\mathrm{Q}}(\boldsymbol{p}) + \max_{j=1,\dots,2^n} f_{\mathrm{S},j}(\boldsymbol{p}), \end{aligned}$$

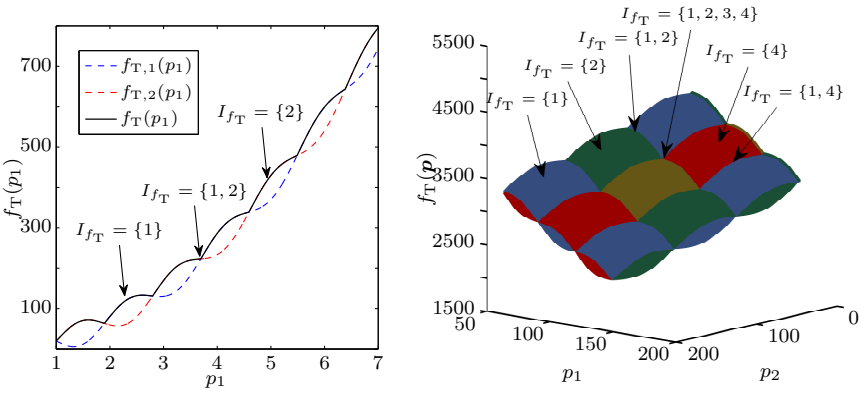
where $a\%b$ denotes the the remainder of the division a/b.

The set of indices $\mathcal{I}_{f_{\mathrm{T}}}(\boldsymbol{p}) = \{j \in \{1, \dots, 2^n\} \mid f_{\mathrm{T}}(\boldsymbol{p}) = f_{\mathrm{T},j}(\boldsymbol{p})\}$ can now be described. Let $\mathcal{S}(\boldsymbol{p})$ be the set of indices of the entries p_i in $\boldsymbol{p}$ that cancel the sine term, and let $\mathcal{F}(\boldsymbol{p})$ be the other indices of $\boldsymbol{p}$:

$$\begin{aligned} \mathcal{S}(\boldsymbol{p}) &= \{s_1, \dots, s_{n_s}\} \\ &= \left\{ i \in \{1, \dots, n\} \mid p_i = p_i^{min} + \frac{k\pi}{e_i},\ k \in \mathbb{N} \right\}, \end{aligned} \tag{7.28}$$

$$\begin{aligned} \mathcal{F}(\boldsymbol{p}) &= \{f_1, \dots, f_{n_f}\} \\ &= \{1, \dots, n\} \setminus \mathcal{S}(\boldsymbol{p}). \end{aligned} \tag{7.29}$$

As depicted in Figure 7.7, the cardinality of $I_{f_{\mathrm{T}}}(\boldsymbol{p})$ is then given by $|I_{f_{\mathrm{T}}}(\boldsymbol{p})| = 2^{n_s}$. If $\mathcal{S}(\boldsymbol{p})$ is empty, only one function $f_{\mathrm{T},j}$ has to be considered, thus $|I_{f_{\mathrm{T}}}(\boldsymbol{p})| = 2^0 = 1$.



(a) $n = 1$ (b) $n = 2$

Figure 7.7: The ELDP cost function $f_{\mathrm{T}}(\boldsymbol{p})$ as the pointwise maximum of 2^n functions $f_{\mathrm{T},j}(\boldsymbol{p})$.

As per (7.25), the generalized projected gradient of f_{T} at $\boldsymbol{p}$ is now defined as follows:

$$\begin{aligned}\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p}) &= \operatorname{co}\{\operatorname{grad} f_{\mathrm{T},j}(\boldsymbol{p}) \mid j \in I_{f_{\mathrm{T}}}(\boldsymbol{p})\} \\ &= P_{\boldsymbol{p}}(\nabla f_{\mathrm{Q}}(\boldsymbol{p})) + \operatorname{co}\{P_{\boldsymbol{p}}(\nabla f_{\mathrm{S},j}(\boldsymbol{p})) \mid j \in I_{f_{\mathrm{T}}}(\boldsymbol{p})\}. \quad (7.30)\end{aligned}$$

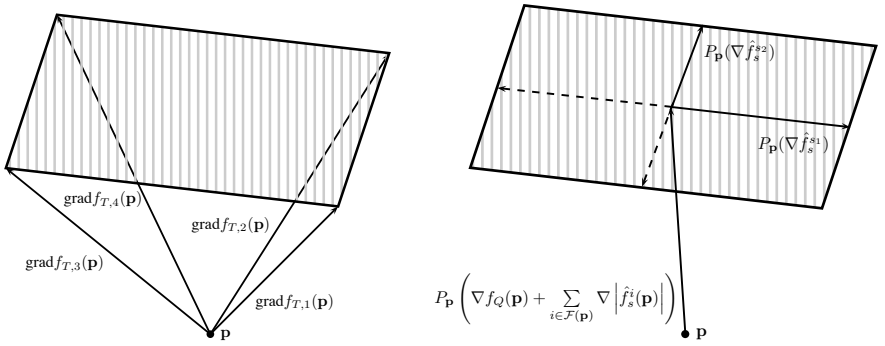
We now consider the quadratic subproblem (7.26) for the ELDP to determine the descent direction $\boldsymbol{d}$ at a point $\boldsymbol{p} \in \mathcal{E}^{n-1}$:

$$\min_{\substack{\lambda_j \geq 0 \\ \sum \lambda_j = 1}} \left\| P_{\boldsymbol{p}} \left(\sum_{j \in \mathcal{I}_{f_{\mathrm{T}}}(\boldsymbol{p})} \lambda_j \nabla f_{\mathrm{T},j}(\boldsymbol{p}) \right) \right\|^2. \quad (7.31)$$

This is a quadratic program with 2^{n_s} variables λ_j. However, as illustrated in Figure 7.8, the number of variables can be drastically reduced since the generalized projected gradient $\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p})$ can actually be described with only n_s vectors. Indeed, noting that $\left|\hat{f}_{\mathrm{s}}^{i}(\boldsymbol{p})\right|$ is differentiable for $i \in \mathcal{F}(\boldsymbol{p})$ and recalling that f_{T} is a separable function, $\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p})$

can be advantageously rewritten in the following way:

$$\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p}) = P_{\boldsymbol{p}}(\nabla f_{\mathrm{Q}}(\boldsymbol{p})) + \sum_{i \in \mathcal{F}(\boldsymbol{p})} P_{\boldsymbol{p}}(\nabla \left| \hat{f}_{\mathrm{S}}^{i}(\boldsymbol{p}) \right|) + \left\{ \sum_{i \in \mathcal{S}(\boldsymbol{p})} \lambda_i P_{\boldsymbol{p}}(\nabla \hat{f}_{\mathrm{S}}^{i}(\boldsymbol{p})) \mid \lambda_i \in [-1, 1] \right\}. \quad (7.32)$$



(a) $\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p})$ described with 2^{n_s} vectors (7.30).

(b) $\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p})$ described with n_s vectors (7.32).

Figure 7.8: Illustration of the generalized gradient (dashed area). The description of $\operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p})$ (7.30) using 2^{n_s} vectors and the description (7.32) using n_s vectors are equivalent.

The quadratic subproblem (7.31) hence becomes:

$$\min_{\lambda_i \in [-1,1]} \left\| P_{\boldsymbol{p}} \left(\nabla f_{\mathrm{Q}}(\boldsymbol{p}) + \sum_{i \in \mathcal{F}(\boldsymbol{p})} \nabla \left| \hat{f}_s^i(\boldsymbol{p}) \right| + \sum_{i \in \mathcal{S}(\boldsymbol{p})} \lambda_j \nabla \hat{f}_s^i(\boldsymbol{p}) \right) \right\|^2. \quad (7.33)$$

Introducing the following notation:

$$\boldsymbol{S} = \left[P_{\boldsymbol{p}}(\nabla \hat{f}_{\mathrm{S}}^{s_1}(\boldsymbol{p})) \ \cdots \ P_{\boldsymbol{p}}(\nabla \hat{f}_{\mathrm{S}}^{s_{n_s}}(\boldsymbol{p})) \right], \quad (7.34)$$

$$\boldsymbol{g} = P_{\boldsymbol{p}} \left(\nabla f_{\mathrm{Q}}(\boldsymbol{p}) + \sum_{i \in \mathcal{F}(\boldsymbol{p})} \nabla \left| \hat{f}_s^i(\boldsymbol{p}) \right| \right), \quad (7.35)$$

the quadratic subproblem can be rewritten as a general bound constrained convex quadratic program of the form:

$$\begin{aligned}\boldsymbol{\lambda}^* &= \arg\min_{\boldsymbol{\lambda}} \ \|\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda}\|^2 \\ &= \arg\min_{\boldsymbol{\lambda}} \boldsymbol{\lambda}^\top \boldsymbol{S}^\top \boldsymbol{S}\boldsymbol{\lambda} + 2\boldsymbol{\lambda}^\top \boldsymbol{S}^\top \boldsymbol{g} \\ &\text{subject to } -1 \leq \boldsymbol{\lambda} \leq 1,\end{aligned} \tag{7.36}$$

where $\boldsymbol{S}^\top\boldsymbol{S}$ is positive semidefinite (positive definite if $\boldsymbol{S}$ has full rank). The descent direction $\boldsymbol{d}$ is then computed as $\boldsymbol{d} = -(\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda}^*)$.

7.4.4 Including the Bound Constraints

Solving the subproblem (7.36) does not always provide an admissible descent direction for points on the boundary of the feasible set Ω. Indeed, if the resulting direction $\boldsymbol{d}$ points outside Ω, no progress can be made by following it. However, the bound constraints can be incorporated into the quadratic problem in order to ensure an admissible descent direction for all points in Ω. Let $l_i(\boldsymbol{p})$ and $u_i(\boldsymbol{p})$, $i = 1, \ldots, n$, denote the lower and upper bound constraints:

$$\begin{aligned} l_i(\boldsymbol{p}) &= -p_i + p_i^{\min} \leq 0, \\ u_i(\boldsymbol{p}) &= p_i - p_i^{\max} \leq 0. \end{aligned}$$

Let $\mathcal{L}(\boldsymbol{p})$ and $\mathcal{U}(\boldsymbol{p})$ be the indices of the active constraints, and let $\mathcal{B}(\boldsymbol{p})$ be their union:

$$\begin{aligned} \mathcal{L}(\boldsymbol{p}) &= \{l_1, \ldots, l_{n_l}\} \\ &= \{i \in \{1, \ldots, n\} \mid l_i(\boldsymbol{p}) = 0\}, \\ \mathcal{U}(\boldsymbol{p}) &= \{u_1, \ldots, u_{n_u}\} \\ &= \{i \in \{1, \ldots, n\} \mid u_i(\boldsymbol{p}) = 0\}, \\ \mathcal{B}(\boldsymbol{p}) &= \mathcal{L}(\boldsymbol{p}) \bigcup \mathcal{U}(\boldsymbol{p}). \end{aligned} \tag{7.37}$$

Noting that $\nabla l_i(\boldsymbol{p}) = -\boldsymbol{e}_i$ and $\nabla u_i(\boldsymbol{p}) = \boldsymbol{e}_i$, we introduce the following notation:

$$\boldsymbol{B} \quad = \quad \left[P_{\boldsymbol{p}}(-\boldsymbol{e}_{l_1}) \ \cdots \ P_{\boldsymbol{p}}(-\boldsymbol{e}_{l_{n_l}}) \mid P_{\boldsymbol{p}}(\boldsymbol{e}_{u_1}) \ \cdots \ P_{\boldsymbol{p}}(\boldsymbol{e}_{u_{n_u}})\right]. \tag{7.38}$$

The search for an admissible descent direction can now include the projected gradient of the active constraints an can thus be expressed as the following quadratic program:

$$(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = \arg\min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \|\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda} + \boldsymbol{B}\boldsymbol{\mu}\|^2 \quad (7.39)$$
$$\text{subject to } \begin{cases} -1 \leq \boldsymbol{\lambda} \leq 1, \\ \boldsymbol{\mu} \geq 0. \end{cases}$$

This is again a convex quadratic problem of dimension $|\mathcal{S}(\boldsymbol{p}) \bigcup \mathcal{B}(\boldsymbol{p})| \leq n$. The steepest-descent admissible direction is finally computed as follows:

$$\boldsymbol{d} = -(\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda}^* + \boldsymbol{B}\boldsymbol{\mu}^*). \quad (7.40)$$

7.4.5 First-Order Stationarity Condition

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a nonsmooth (Lipschitz) function to be minimized subject to a set of constraints $c_i(\boldsymbol{x}) \leq 0$ $(i = 1, \ldots, m)$. The first order necessary stationarity conditions for this constrained nonsmooth problem can be stated as follows (see *e.g.* [Cla76], Section 3):

Theorem 7.1. *If $\boldsymbol{x}^*$ minimizes $f(\boldsymbol{x})$ locally, then there exist numbers μ_i $(i = 1, \ldots, m)$ such that:*

(a) $\mu_i \geq 0$,

(b) $\mu_i c_i(\boldsymbol{x}^*) = 0$,

(c) $\mathbf{0} \in \partial f(\boldsymbol{x}^*) + \sum_{i=1}^m \mu_i \nabla c_i(\boldsymbol{x}^*)$.

Applying this theorem to the ELDP, we obtain the following characterization of its local minimizers:

Theorem 7.2. *If $\boldsymbol{p}^*$ minimizes $f_{\mathrm{T}}(\boldsymbol{p})$ locally in Ω, then there exist numbers μ_i^-, μ_i^+ $(i = 1, \ldots, n)$ such that:*

(a) $\mu_i^- \geq 0$, $\mu_i^+ \geq 0$,

(b) $\mu_i^- l_i(\boldsymbol{p}^*) = 0$, $\mu_i^+ u_i(\boldsymbol{p}^*) = 0$,

(c) $\mathbf{0} \in \operatorname{grad} f_{\mathrm{T}}(\boldsymbol{p}^*)$
$+ \sum_{i=1}^n \left(\mu_i^- \operatorname{grad}(l_i(\boldsymbol{p}^*)) + \mu_i^+ \operatorname{grad}(u_i(\boldsymbol{p}^*))\right)$.

The technique proposed in the previous section to compute admissible descent directions offers an important advantage over the classical alternatives (barrier term, augmented Lagrangian, etc.). Computing the direction $\boldsymbol{d} = -(\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda}^* + \boldsymbol{B}\boldsymbol{\mu}^*)$ at a local minimizer point $\boldsymbol{p}^*$ yields the zero vector. Therefore, the norm of the direction $\boldsymbol{d}$ can be used as a stopping criterion for the descent algorithm, ensuring that it will stop at a (numerically) stationary point.

7.5 Implementation

In the previous sections, all the ingredients needed to perform local optimization of $f_{\mathrm{T}}(\boldsymbol{p})$ in Ω were introduced. In this section, we present how, given a feasible initial iterate $\boldsymbol{p}^0 \in \Omega$, the Riemannian subgradient descent method is applied to produce a sequence of iterates $\boldsymbol{p}^k \in \Omega$. The implementation details are now given.

7.5.1 Generating a feasible iterate

Recalling the geometrical considerations presented in Section 7.2.2, the feasible set Ω is the intersection of an axis-aligned box and an ellipsoid surface. From there, a simple approach to generate a feasible point $\boldsymbol{p} \in \Omega$ is given by Algorithm 7.1. The loop is needed because the scaling performed by R may output a point outside the bounds constraints. Note that Algorithm 7.1 becomes superfluous if the proposed local method is combined with a feasible global method (i.e, with iterates on Ω).

Algorithm 7.1 Feasible point generation

Output: $\boldsymbol{p} \in \Omega$, a feasible point
  repeat
    Draw $\boldsymbol{x}$ from the uniform distribution on
$$[p_1^{\min}, p_1^{\max}] \times \cdots \times [p_n^{\min}, p_n^{\max}]$$
    $\boldsymbol{p} \leftarrow R_{\boldsymbol{x}}(\boldsymbol{0})$, where R is given by formula (7.15)
  until $\boldsymbol{p} \in \Omega$
  return $\boldsymbol{p}$

Note that generating a feasible iterate can be NP-Hard and therefore, using the aforementioned algorithm to generate one might prove very inefficient depending on the geometrical configuration of the instance of

the problem at hand (for instance if the intersection of the box and the ellipsod depicted in Figure 7.1 is very small).

7.5.2 Computing the descent direction

The steepest-descent admissible direction at a point $\boldsymbol{p}^k$ is described by (7.40). It can be computed, provided that the sets $\mathcal{S}(\boldsymbol{p}^k)$ and $\mathcal{B}(\boldsymbol{p}^k)$ are available. However, in practice, it is numerically unlikely for an iterate $\boldsymbol{p}^k$ to ever cancel exactly any of the sine terms in $f_{\mathrm{T}}(\boldsymbol{p}^k)$. Therefore, $\mathcal{S}(\boldsymbol{p}^k)$ and its complement $\mathcal{F}(\boldsymbol{p}^k)$ are approximated in the following way:

$$\mathcal{S}_\epsilon(\boldsymbol{p}) \quad = \quad \left\{ i \in \{1, \ldots, n\} : \left| p_i - \left(p_i^{\min} + \frac{k\pi}{e_i} \right) \right| \leq \epsilon \right\}, \quad (7.41)$$

$$\mathcal{F}_\epsilon(\boldsymbol{p}) = \{1, \ldots, n\} \setminus \mathcal{S}_\epsilon(\boldsymbol{p}). \quad (7.42)$$

Similarly, the bound constraints indices are approximated as follows:

$$\mathcal{B}_\epsilon(\boldsymbol{p}) = \{ i \in \{1, \ldots, n\} \mid l_i(\boldsymbol{p}) \geq -\epsilon \text{ or } u_i(\boldsymbol{p}) \geq -\epsilon \}. \quad (7.43)$$

An "ϵ–steepest-descent ϵ-admissible direction" is then computed at each iteration, using Algorithm 7.2. In its nondifferentiable-related aspect, this approach is reminiscent of Goldstein's ϵ-generalized gradient [Gol77]. In its bound-related aspect, it has an active-set flavour.

Algorithm 7.2 DescentDirection$(\boldsymbol{p}, \mathcal{S}_\epsilon(\boldsymbol{p}), \mathcal{F}_\epsilon(\boldsymbol{p}), \mathcal{B}_\epsilon(\boldsymbol{p}))$ - Descent direction

Input: $\boldsymbol{p} \in \Omega$, $\mathcal{S}_\epsilon(\boldsymbol{p})$, $\mathcal{F}_\epsilon(\boldsymbol{p})$, $\mathcal{B}_\epsilon(\boldsymbol{p})$
Output: $\boldsymbol{d}$, an admissible descent direction
Compute $\boldsymbol{S}$, $\boldsymbol{g}$ and $\boldsymbol{B}$ at $\boldsymbol{p}$ according to (7.34), (7.35) and (7.38), in which $\mathcal{S}(\boldsymbol{p})$, $\mathcal{F}(\boldsymbol{p})$, $\mathcal{B}(\boldsymbol{p})$ are replaced by $\mathcal{S}_\epsilon(\boldsymbol{p})$, $\mathcal{F}_\epsilon(\boldsymbol{p})$, $\mathcal{B}_\epsilon(\boldsymbol{p})$
Compute $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solving the quadratic programming subproblem (7.39)
$\boldsymbol{d} \leftarrow -(\boldsymbol{g} + \boldsymbol{S}\boldsymbol{\lambda}^* + \boldsymbol{B}\boldsymbol{\mu}^*)$
return $\boldsymbol{d}$

7.5.3 Computing the step size

Once a descent admissible direction has been computed, the step size must be determined. We propose to adopt Armijo's rule with two minor

modifications. First, the initial step size is retained from the previous step size computation. Second, before performing classical backtracking, a few trials are performed by expanding the previous step size. These two modifications provide possible faster convergence of the descent algorithm by allowing bigger step sizes. The line search uses the retraction $\hat{R}$ defined by (7.19). In practice, given a vector $\boldsymbol{\xi}$, the corresponding set $\mathcal{C}$ involved in (7.19) is approximated as follows:

$$\mathcal{C}_\epsilon = \{i \in \{1, \dots, n\} \mid |\boldsymbol{\xi}_i| \leq \epsilon\}. \tag{7.44}$$

The computation of the step size is described in Algorithm 7.3.

Algorithm 7.3 StepSize$(\boldsymbol{p}, \boldsymbol{g}, \alpha_0, \beta, \gamma, n_\mathrm{f}, n_\mathrm{b})$ - Line search using Armijo's rule

Input: $\boldsymbol{p} \in \Omega$, $\boldsymbol{g}$ search direction, α_0 initial step size, β sufficient decrease constant, γ step size shrinking factor, n_f number of forward steps, n_b number of backtracking steps

Output: α, a step size ensuring sufficient decrease

Consider $\tilde{f}_\mathrm{T}(\boldsymbol{p}) = \begin{cases} f_\mathrm{T}(\boldsymbol{p}) & \text{if (7.2)}, \\ +\infty & \text{otherwise.} \end{cases}$

Step 1 (Forward exploration):

```
for s = n_f → 1 do
    if f̃_T(R̂_p(γ^{-s} g/‖g‖)) < f_T(p) − βα_0γ^{-s} ‖g‖ then
        return α_0γ^{-s}
    end if
end for
```

Step 2 (Backtracking):

```
for s = 0 → n_b do
    if f̃_T(R̂_p(γ^s g/‖g‖)) < f_T(p) − βα_0γ^s ‖g‖ then
        return α_0γ^s
    end if
end for
return 0
```

7.5.4 Subgradient Descent for the ELDP

The subgradient descent algorithm can now be completely described and is given by Algorithm 7.4. In the taxonomy of optimization meth-

ods [NW06], the proposed approached can be considered an active-set–like feasible ϵ-subgradient–descent method for nonsmooth nonconvex problems with both equality and bound constraints.

Algorithm 7.4 Riemannian Subgradient Descent for the ELDP

Input: $\boldsymbol{p}^0 \in \Omega$, ϵ singular point detection tolerance, δ descent direction norm tolerance, β sufficient decrease constant, γ step size shrinking factor, n_f number of forward steps, n_b number of backtracking steps
Output: $\boldsymbol{p}^* \in \Omega$, a generalized stationary point
Set $k = 0$, $\alpha^0 = 1$
Step 1 (Compute the admissible descent direction)
Compute $\mathcal{S}_\epsilon(\boldsymbol{p}^k)$, $\mathcal{F}_\epsilon(\boldsymbol{p}^k)$ and $\mathcal{B}_\epsilon(\boldsymbol{p}^k)$ (7.41)–(7.43)
$\boldsymbol{d}^k \leftarrow \textsc{DescentDirection}(\boldsymbol{p}^k, \mathcal{S}_\epsilon(\boldsymbol{p}^k), \mathcal{F}_\epsilon(\boldsymbol{p}^k), \mathcal{B}_\epsilon(\boldsymbol{p}^k))$
if $\left\|\boldsymbol{d}^k\right\| < \delta$ **then**
return $\boldsymbol{p}^k$
end if
$\boldsymbol{g}^k \leftarrow \boldsymbol{d}^k / \left\|\boldsymbol{d}^k\right\|$
Step 2 (Compute the step size)
$\alpha^k \leftarrow \textsc{StepSize}(\boldsymbol{p}^k, \boldsymbol{g}^k, \alpha^{k-1}, \beta, \gamma, n_\mathrm{f}, n_\mathrm{b})$
Step 3 (Compute the new iterate)
if $\alpha^k > 0$ **then**
$\boldsymbol{p}^{k+1} \leftarrow \hat{R}_{\boldsymbol{p}^k}(\alpha^k \boldsymbol{g}^k)$
$k \leftarrow k + 1$
Go to **Step 1**
else
return $\boldsymbol{p}^k$
end if

7.6 Numerical Experiments

The Riemannian subgradient descent algorithm presented in the previous section was implemented in MATLAB. The data from four instances of ELDP with valve-point effect were collected, for dimensions $n = 3,\ 5,\ 6$ and 15 (sources: [Gai03], [MuAWA10], [Gai04], [SHP$^+$10]).

These data sets contain the following information for each instance:

- the coefficients of f_{T} for the quadratic term:

$$a_i \ [\$/(h.\mathrm{MW}^2)],\ b_i \ [\$/(h.\mathrm{MW})],\ c_i \ [\$/h],$$

- the coefficients of f_{T} for the sine term:

$$d_i \ [\$/h],\ e_i \ [1/\mathrm{MW}],$$

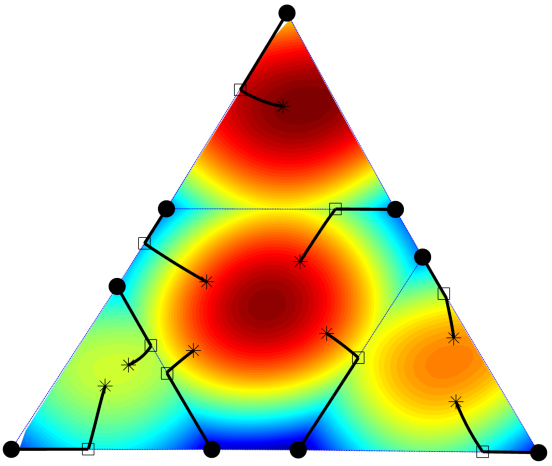
- the bound constraints: $p_i^{\min}$, $p_i^{\max}$ [MW],
- the power demand: p_d [MW],
- the transmission loss coefficients:

$$\boldsymbol{B} \ [1/\mathrm{MW}],\ \boldsymbol{B}^0 \ [-],\ B^{00} \ [\mathrm{MW}].$$

An important remark about these data sets is that for the cases $n = 5$ and $n = 6$, only the first two dimensions present the valve-point effect ($d_i = e_i = 0,\ i \geq 2$). The consequence is that the multimodal aspect for these two data sets is much less pronounced compared to the the two other data sets, for which the valve-point effect is present in all dimensions.
The MATLAB code and data sets are available online at `http://sites.uclouvain.be/absil/borckmans/ELDP`. In the following experiments, the default parameters were chosen as follows: $\epsilon = 10^{-8}$, $\beta = 10^{-4}$, $\gamma = 0.5$, $n_{\mathrm{f}} = 3$, $n_{\mathrm{b}} = 50$, $\delta = 10^{-12}$.

7.6.1 Local Convergence

The trajectories generated by Algorithm 7.4 for different starting points are illustrated in Figure 7.9 for the case $n = 3$. Whenever the sequence of iterates approaches a nondifferentiable point, the projected subgradients are computed, and the descent direction resulting from the quadratic subproblem follows the corresponding sub-ellipsoid. These sub-ellipsoids are represented as dashed lines in Figure 7.9. For comparison purpose, applying a simple Riemannian gradient descent (ignoring the nondifferentiable aspect of f_{T}) from the same starting points yields convergence to nonstationary points. Algorithm 7.4 was tested by performing a large number of trials for each ELDP instance, starting from random points.



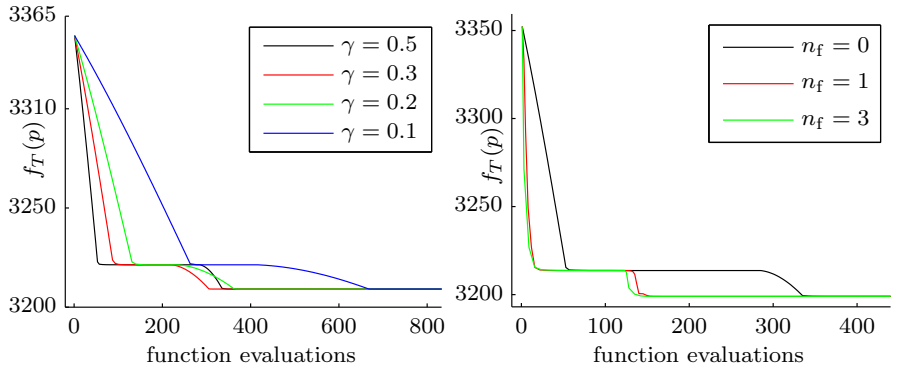
(a) Local minima for $n = 3$

Figure 7.9: Local convergence illustration for $n = 3$. The trajectories from 9 starting points (stars) are depicted. The final iterates obtained using a naive steepest-descent algorithm (squares) and the proposed Riemannian subgradient algorithm (dots) are represented. The dashed blue lines correspond to the nondifferentiable sub-ellipsoids.

The final iterates of the trials were tested for the necessary stationarity condition; the norm of the final descent direction, as computed by Algorithm 7.2 was smaller than 10^{-12} for all the trials. Considering two points to be different if their corresponding costs are at least 10^{-8} apart from each other, the proposed algorithm was able to consistently converge to a small set of potential local minima for the cases $n = 3$, 5 and 6 (9 minima for $n = 3$, 7 minima for $n = 5$, and 7 minima for $n = 6$). As mentioned earlier, the small number of local minima for the cases $n = 5$ and $n = 6$ is due to the limited valve-point effect. On the other hand, for the case $n = 15$, the number of local minima is much higher since the valve-point effect affects all the dimensions. After 10^6 trials, the proposed approach consistently identified solutions in a set of around 2×10^5 points.

7.6.2 Parameter Influence

As depicted in Figure 7.10(a) and 7.10(b), the convergence speed is affected by the line search scaling factor γ and is enhanced when reusing



(a) Influence of the scaling factor γ (b) Influence of forward exploration

Figure 7.10: Local convergence properties for $n = 3$.

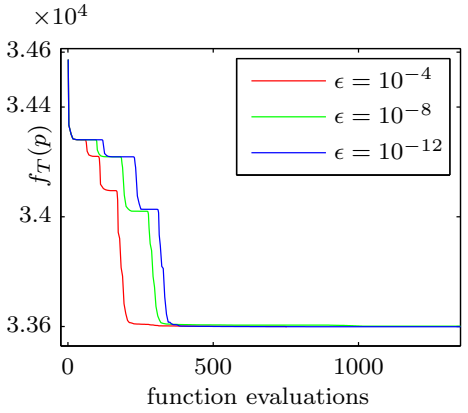
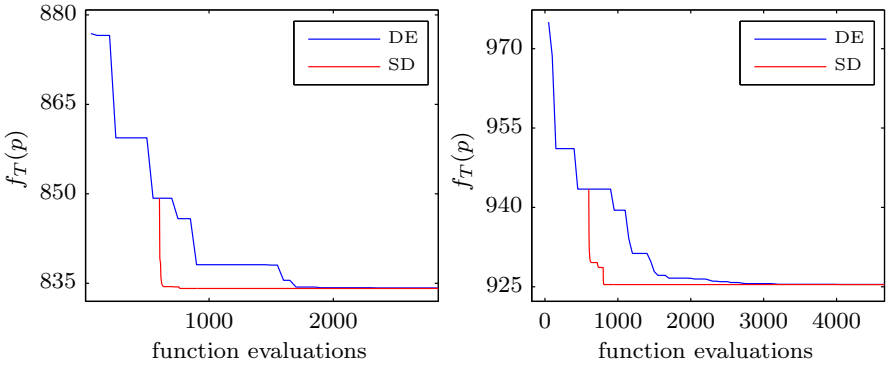


Figure 7.11: Influence of ϵ ($n = 15$)

the previous successful step size and allowing forward exploration (n_f steps).

Another important parameter of the proposed algorithm is ϵ, the tolerance for detecting the singular entries in (7.41). Smaller values for ϵ lead to better precision but at the cost of taking more function evaluations to detect singularities. This can be observed in Figure 7.11, where the plateaus correspond to iterates approaching singular sub-ellipsoids of Ω.



(a) $n = 5$ (b) $n = 6$

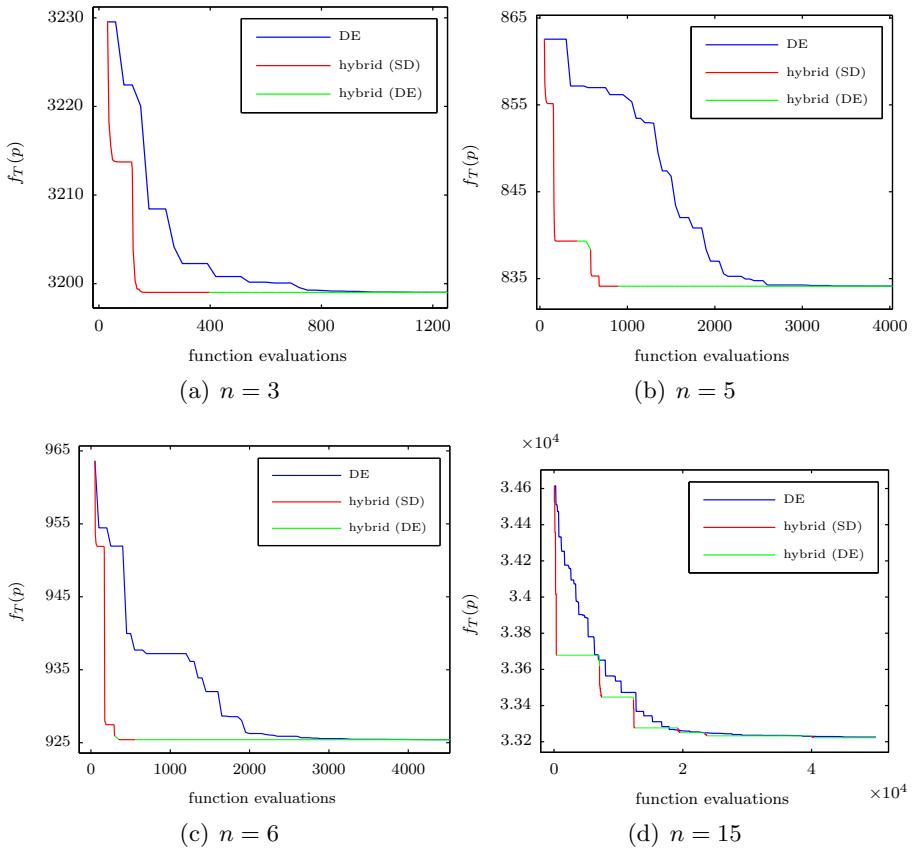
Figure 7.12: Using the proposed subgradient descent (SD) algorithm as a post processing tool. Stopping the DE algorithm progression at some point and refining the output using the proposed SD algorithm (red curve) provides faster convergence than DE alone (blue curve).

7.6.3 Global Exploration

The Riemannian subgradient descent presented in this chapter is inherently a local optimizer. It is very efficient at computing a local minimizer in the vicinity of a given starting point in Ω, but it is not meant to roam around the cost function landscape in search of the global minimum. On the other hand, the global heuristics that are largely explored in the literature have very complementary properties: they often provide good exploration of the search space but they tend to struggle to refine the final iterate and they do not attempt to check stationarity (and even feasibility in some cases) of their output. This calls for hybrid algorithms that combine the best of both approaches.

In order to illustrate this idea, we implemented the Differential Evolution (DE) algorithm as proposed in [BC10]. This population-based heuristic is known to provide reasonably good exploration of the search space, relying quite heavily on random updates, combining selected individuals. As in many heuristics proposed to solve the ELDP, the authors of [BC10] use the concept of a slack variable to respect the equality constraint (7.3).

A first approach to hybridize DE with our algorithm, known as sequential approach [Sel11], is to stop the global exploration at some point before applying the subgradient descent (Algorithm 7.4). This post-



(a) $n = 3$ (b) $n = 5$

(c) $n = 6$ (d) $n = 15$

Figure 7.13: Benefit of hybridizing DE with the subgradient descent (SD) algorithm. In this cyclical hybrid approach, DE is used to escape local minima (green curve), then the proposed SD algorithm is applied to refine the best individual (red curve). The outcome is a faster convergence than when the DE technique is applied alone (blue curve).

processing step avoids the slow refinement of the solution and produces in practice a final iterate that satisfies the stationary conditions from Theorem 7.2. The typical behaviour of such an approach is depicted in Figure 7.12.

Another option is to integrate the subgradient descent within the global technique, following the cyclical approach mentioned in [Sel11]. The idea is to refine the best individual produced by the DE algorithm at each iteration using the projected subgradient descent. The benefit

of this approach is illustrated in Figure 7.13.

7.7 Conclusion

In this chapter, we showed how the geometrical structure of the Economic Load Dispatch Problem (7.5) can be exploited to perform local optimization of its associated cost function. The canvas of optimization on Riemannian manifolds was used to maintain feasible iterates at all times. The nondifferentiable nature of the cost function was handled using Clarke's generalized calculus. A simple procedure was presented to compute an admissible descent direction and to perform a curvilinear search along that direction while satisfying the constraints. The resulting local method is unique in that it possesses the combination of desirable properties mentioned in the introduction. Finally, we showed that the multimodal aspect of the ELDP can be addressed by hybridizing the proposed local method with a global optimization technique. In further works, it would be interesting to conduct a complete convergence analysis of the proposed algorithms.

Closing

Chapter 8

Conclusion and perspectives

We started this manuscript with a rather obvious statement: optimization is everywhere. Whether applied as a tool by the pragmatical engineer, studied in depth by a mathematician, or used unconsciously by the end-user of a high-tech gadget, optimization is indeed prominent in a tremendous number of activities. In the light of the material we studied during our work, another important feature appears: optimization is an art and there exist numerous ways to approach it. As with many artistic areas, different trends coexist and their respective followers tend to praise their own vision of the field. On one side, theoreticians study the foundations of the art in a meticulous way and provide a robust basis on which everyone else can rely. On the other side, practicians come up with original algorithmic ideas, sometimes without giving them a careful thought. Greatness in art often originates from the fusion of different techniques and genres. The same probably holds for optimization.

In this thesis, we studied applications originating from various engineering fields. These applications were used as models, exhibiting complex properties, that forced us to discover, learn and apply different fields of optimization. This process is far from being a straight path, but it is a really enriching one. However, as in any creative process, it is common to have a feeling of unachieved work. And often for good reasons. As a palliative to this situation, we now give a summary of the main contributions of this thesis, while highlighting the points for which there is room for more work or, in some cases, could have been investigated

under a completely different angle.

Main contributions

As already stated in the introduction of this thesis, while the applications are at the center of the developments, their purpose is above all to provide concrete instances of problems with given characteristics. In other words, although it would have been possible to reformulate some of these applications (for instance to make their objective function convex), we intentionally worked in their original settings. The overall intent was to provide proof-of-concept solutions to deal with nonsmooth and multimodal problems on matrix manifold rather than to provide finely tuned methods achieving the best numerical results for the applications at hand. This also explains why, in the applicative chapters, the emphasis was often put on the development and adaptation of the methods rather than on extensive numerical experiments.

With the first application—the oriented bounding box problem (Chapter 4)—we showed how metaheuristics techniques can be extended to work in Riemannian manifolds settings. While these heuristics do not come with detailed convergence proofs, they perform well on multimodal landscape and allow one to obtain satisfactory solutions easily. The algorithms proposed in this chapter are based on the simplest versions of the heuristics (PSO and DE) and could benefit from many variations of these techniques to provide, among other aspects, better exploration of the search space and/or the ability to identify several local minima (e.g. with niching techniques). Following the guidelines established in this chapter, these extensions are expected to be straightforward, as was demonstrated in the second applicative chapter, with the GCPSO variation of the original PSO algorithm.

With the second application—the higher-order tensor approximation problem (Chapter 5)—we took this process a step further. We reused the PSO algorithm, but this time including more sophisticated mechanisms to ensure local convergence. Furthermore, the PSO algorithm was adapted to the Cartesian product of a different manifold: the Grassmann manifold. The ability of this adapted GCPSO algorithm to discover local minima from a multimodal landscape was demonstrated.

We approached the third application—the range-based independent component analysis problem (Chapter 6)—with a different perspective, shifting from the heuristics world to a derivative-free technique: the

mesh adaptive direct search algorithm (MADS). We were able to adapt this algorithm to a spherical setting (and further to the Oblique manifold) while preserving its strong convergence properties. Although this chapter was dedicated to a specific application (range-based ICA), the adaptation of the MADS algorithm to a manifold setting is highly generalizable and can be beneficial for a broad variety of problems, exhibiting non-differentiable objective functions.

In the last application—the economic load dispatch problem (Chapter 7)—we once more changed our working framework. While the ELDP problem is clearly suitable for the approaches used in the previous chapters (heuristics and derivative-free techniques), this time we wanted to exploit the properties of the associated cost function and optimization landscape as much as possible. We proposed a Riemannian subgradient descent algorithm, able to construct feasible descent directions at all times, by solving simple, low-dimensional quadratic programs.

Future work and perspectives

The heuristics that were used in the two first applications both involve some stochastic components. While they are certainly helpful to perform global exploration of the search space, this randomness also makes it complicated to build in-depth convergence analyses. Another important characteristic of metaheuristics is the importance of parameters adjustment. Although it is a common trend in this field to devote an important part of the research to fine tuning and analysis of parameters influence, this was not the case in our research work as we mainly focused on the development and adaptation of the methods themselves. An important ongoing research topic is therefore to study the importance of these parameters, possibly with the help of techniques allowing automatic and/or dynamical parameter adjustment.

The variety of methods studied during our research represent only a small portion of the optimization techniques available nowadays. However, the mechanisms presented in the four applicative chapters to adapt existing methods to manifold settings are often general enough to be reusable. In this spirit, the development of a Matlab toolbox for optimization on manifolds started a few months ago in our research group. The intent of this toolbox is to allow the development of new methods and/or geometries in a user-friendly way (more information on Manopt is available at `http://www.manopt.org`). By making common geome-

tries and well studied solvers readily available, such a toolbox allows one to push numerical experiments further, while making comparisons and benchmarking much easier than having to manually and individually implement the methods and geometries.

As could be observed in the last applicative chapter, the hybridization of optimization methods often brings higher quality solutions. While the concept of hybridization is definitely not new, it probably deserves a better understanding from the scientific community. Mixing well known algorithms, equipped with strong convergence analysis, with more exotic search heuristics represents an interesting research path and should not be disregarded because of the lack of available theory as was the case for instance with derivative-free optimization techniques in the past.

Finally, optimization is nowadays a very active area of research, and can be expected to remain so in the long term. Which trend will take the lead in this broad community is hard to say for the moment, but we will take the bet that optimization as a field will benefit from successful interactions and collaborations between different communities.

Bibliography

[AADD09] Mark A. Abramson, Charles Audet, J. E. Dennis, and Sébastien Le Digabel. Orthomads: A deterministic mads instance with orthogonal directions. *SIAM J. on Optimization*, 20(2):948–966, July 2009.

[ABB+07] E. Acar, C. A. Bingol, H. Bingol, R. Bro, and B. Yener. Multiway analysis of epilepsy tensors. *ISMB 2007 Conference Proceedings, Bioinformatics*, 23(13):i10–i18, 2007.

[ACD07] Charles Audet, A L Custodio, and J E Jr Dennis. Mesh adaptive direct search algorithms for constrained optimization (vol 17, pg 188, 2006). *SIAM Journal on Optimization*, 18(4):1501–1503, 2007.

[AD02] Charles Audet and J. E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM J. on Optimization*, 13(3):889–903, August 2002.

[ADM+02] Roy L. Adler, Jean-Pierre Dedieu, Joseph Y. Margulies, Marco Martens, and Mike Shub. Newton's method on Riemannian manifolds and a geometric model for the human spine. *IMA J. Numer. Anal.*, 22(3):359–390, July 2002.

[AG06] P A Absil and K A Gallivan. Joint diagonalization on the oblique manifold for independent component analysis. In *In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP*, pages 945–948, 2006.

[AKTH02] P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa. A hybrid EP and SQP for dynamic economic dispatch with nonsmooth fuel cost function. *Power Engineering Review, IEEE*, 22(4):77, april 2002.

[AM12] P.-A. Absil and Jérôme Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[AMS04] P.-A. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Appl. Math.*, 80(2):199–220, January 2004.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[ASAO10] J.S. Alsumait, J.K. Sykulski, and A.K. Al-Othman. A hybrid GA–PS–SQP method to solve power system valve-point economic dispatch problems. *Applied Energy*, 87(5):1773 – 1781, 2010.

[ASDM04] P. A. Absil, R. Sepulchre, P. Van Dooren, and R. Mahony. Cubically convergent iterations for invariant subspace computation. *SIAM J. Matrix Anal. Appl.*, 26(1):70–96, 2004.

[BA10] Pierre B. Borckmans and Pierre-Antoine Absil. Oriented bounding box computation using particle swarm optimization. In *Proceedings of the 18th European Symposium on Artificial Neural Networks*, ESANN 2010, 2010.

[BA11] N. Boumal and PA Absil. Rtrmc: A riemannian trust-region method for low-rank matrix completion. *Advances in Neural Information Processing Systems*, 24:406–414, 2011.

[BC10] A. Bhattacharya and P.K. Chattopadhyay. Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch. *IEEE Transactions on Power Systems*, 25(4):1955 –1964, nov. 2010.

[BCA13] S. Easter Selvan Pierre B. Borckmans, Amit Chattopadhyay, and P.-A. Absil. Spherical mesh adaptive direct search for separating quasi-uncorrelated sources by range-based independent component analysis. *Neural Comput.*, 2013. accepted.

[BEvdB02] R. Brits, A. Engelbrecht, and F. van den Bergh. A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 2, pages 692–696, 2002.

[BHP99] Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 82–91, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[BIA10] Pierre B. Borckmans, Mariya Ishteva, and Pierre-Antoine Absil. A modified particle swarm optimization algorithm for the best low multilinear rank approximation of higher-order tensors. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 13–23, Berlin, Heidelberg, 2010. Springer-Verlag.

[BLO05] James V. Burke, Adrian S. Lewis, and Michael L. Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.*, 15(3):751–779, 2005.

[Boo03] William M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, 2003. Revised Second Edition.

[BPR04] R. Boscolo, H. Pan, and V. P. Roychowdhury. Independent component analysis based on nonparametric density estimation. *Trans. Neur. Netw.*, 15(1):55–65, January 2004.

[Bro65] C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19(92):577–593, 1965.

[Bro72] R. W. Brockett. System theory on group manifolds and coset spaces. *SIAM Journal on Control*, 10:265–0, May 1972.

[BSBA13] Pierre B. Borckmans, S. Easter Selvan, Nicolas Boumal, and P.-A. Absil. A Riemannian subgradient algorithm for economic dispatch with valve-point effect. *J. Comput. Applied. Math.*, 2013. accepted.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[CC03] Nareli Cruz Cortés and Carlos A. Coello Coello. Multiobjective optimization using ideas from the clonal selection principle. In *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartI*, GECCO'03, pages 158–170, Berlin, Heidelberg, 2003. Springer-Verlag.

[CGM11] Chia-Tche Chang, Bastien Gorissen, and Samuel Melchior. Fast oriented bounding box optimization on the rotation group so(3,ℝ). *ACM Trans. Graph.*, 30(5):122:1–122:16, October 2011.

[CGT00] A.R. Conn, N.I.M. Gould, and P.L. Toint. *Trust-region methods*, volume 1. Society for Industrial Mathematics, 2000.

[Cha08] Uday K. Chakraborty. *Advances in Differential Evolution*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[Chi92] Alpha C. Chiang. *Elements of dynamic optimization / Alpha C. Chiang*. McGraw-Hill, New York, 1992.

[Cla75] Frank H. Clarke. Generalized gradients and applications. *Transactions of the American Mathematical Society*, 205:pp. 247–262, 1975.

[Cla76] F. H. Clarke. A new approach to Lagrange multipliers. *Mathematics of Operations Research*, 2:165–174, 1976.

[CLL+12] Jiejin Cai, Qiong Li, Lixiang Li, Haipeng Peng, and Yixian Yang. A hybrid CPSO–SQP method for economic dispatch considering the valve-point effects. *Energy Conversion and Management*, 53(1):175 – 181, 2012.

[CP00] I. D. Coope and C. J. Price. Frame-Based Methods for Unconstrained Optimization. *Journal of Optimization Theory and Applications*, 107(2):261–274, 2000.

[CS93] J. F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140(6):362–370, 1993.

[CS03] Yann Collette and Patrick Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer, 2003.

[CSV09a] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[CSV09b] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[DAK00] S. C. Douglas, S. Amari, and S. Y. Kung. On gradient adaptation with unit-norm constraints. *Signal Processing, IEEE Transactions on*, 48(6):1843–1847, June 2000.

[DBDA09] Swagatam Das, Arijit Biswas, Sambarta Dasgupta, and Ajith Abraham. Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications. *Foundations of Computational ...*, 2009.

[DDV00a] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, April 2000.

[DDV00b] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, April 2000.

[DHL07] Gunther Dirr, Uwe Helmke, and Christian Lageman. Nonsmooth riemannian optimization with applications to sphere packing and grasping. In F. Allgüwer, P. Fleming, P. Kokotovic, A.B. Kurzhanski, H. Kwakernaak, A. Rantzer, J.N. Tsitsiklis, Francesco Bullo, and Kenji Fujimoto, editors, *Lagrangian and Hamiltonian Methods for Nonlinear Control 2006*, volume 366 of *Lecture Notes in Control and Information Sciences*, pages 29–45. Springer Berlin / Heidelberg, 2007.

[DKKR09] Darko Dimitrov, Christian Knauer, Klaus Kriegel, and Gunter Rote. Bounds on the quality of the pca bounding boxes. *Computational Geometry*, 42(8):772 – 789, 2009. Special Issue on the 23rd European Workshop on Computational Geometry.

[Dre07] D.W̃. Dreisigmeyer. Direct search algorithms over riemannian manifolds. *Los Alamos National Laboratory. Los Alamos, NM. [Online]. Available: http://www.optimization-online.org/DB_HTML/2007/08/1742.html*, 2007.

[DV04] L. De Lathauwer and J. Vandewalle. Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_N)$ reduction in multilinear algebra. *Linear Algebra Appl.*, 391:31–55, November 2004. Special Issue on Linear Algebra in Signal and Image Processing.

[EAS98] Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, 1998.

[Eng06] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[Eri04] Christer Ericson. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[Feo06] Vitaliy Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[Gai03] ZL Gaing. Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Transactions on Power Systems*, 18(3):1187–1195, AUG 2003.

[Gai04] Z.L. Gaing. Closure to discussion of particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Transactions on Power Systems*, 19(4):2122–2123, 2004. cited By (since 1996) 8.

[Gol77] A. A. Goldstein. Optimization of Lipschitz continuous functions. *Math. Programming*, 13(1):14–22, 1977.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[Hit27a] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematical Physics*, 6(1):164–189, 1927.

[Hit27b] F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematical Physics*, 7(1):39–79, 1927.

[HKO+01] A. Hyv&, J. Karhunen, E. Oja, et al. *Independent component analysis*. Wiley-Interscience, 2001.

[HL09] Christopher J. Hillar and Lek-Heng Lim. Most tensor problems are np hard. *CoRR*, abs/0911.1393, 2009.

[HM94] Uwe Helmke and John B. Moore. *Optimization and Dynamical Systems*. Communications and Control Engineering Series. Springer-Verlag London Ltd., London, 1994. With a foreword by R. Brockett.

[Hyv99a] A. Hyvärinen. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

[Hyv99b] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[Hyv12] A. Hyvärinen. Independent component analysis: Recent advances. In *Philosophical Transactions of the Royal Society A*, 2012. In press.

[IAHL11] Mariya Ishteva, Pierre-Antoine Absil, Sabine Van Huffel, and Lieven De Lathauwer. Best low multilinear rank approximation of higher-order tensors, based on the riemannian trust-region scheme. *SIAM J. Matrix Analysis Applications*, 32(1):115–135, 2011.

[IAVD10] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer. Tucker compression and local optima. Technical Report 09-247, ESAT-SISTA, K.U.Leuven, Belgium, 2010.

[Ish09] M. Ishteva. *Numerical methods for the best low multilinear rank approximation of higher-order tensors*. PhD thesis, Department of Electrical Engineering, Katholieke Universiteit Leuven, December 2009.

[Jou09] Michel Journée. *Geometric algorithms for component analysis with a view to gene expression data analysis*. PhD thesis, Université de Liège, 2009.

[KA09] Dervis Karaboga and Bahriye Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1):61–85, June 2009.

[Kd80] P. M. Kroonenberg and J. de Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, 1995.

[KLT03] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45:385–482, 2003.

[KN96] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry*. Number vol. 2 in Wiley Classics Library. Wiley, 1996.

[KOW+97] J. Karhunen, E. Oja, L. Wang, R. Vigário, and J. Joutsensalo. A class of neural networks for independent component analysis. *IEEE Transactions on Neural Networks*, 8(3):486–504, May 1997.

[Kro08] P. M. Kroonenberg. *Applied Multiway Data Analysis*. Wiley, 2008.

[LA10] Xi-Lin Li and Tülay Adali. Complex independent component analysis by entropy bound minimization. *Trans. Cir. Sys. Part I*, 57(7):1417–1430, July 2010.

[Lag07] C. Lageman. *Convergence of Gradient-like Dynamical Systems and Optimization Algorithms*. PhD thesis, Julius-Maximilians-Universität Würzburg, 2007.

[Lee03] John M. Lee. *Introduction to smooth manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2003.

[LSG04] Xiuwen Liu, Anuj Srivastava, and Kyle Gallivan. Optimal linear representations of images for object recognition. *IEEE Pattern Anal. and Mach. Intell.*, 26(5):662–666, May 2004.

[Lue72] David G. Luenberger. The gradient projection method along geodesics. *Management Science*, 18(11):620–631, 1972.

[LVV06] John A. Lee, Frédéric Vrins, and Michel Verleysen. Non-orthogonal support width ica. In *In Proc. ESANN 2006*, 2006.

[LVV08] John A. Lee, Frédéric Vrins, and Michel Verleysen. Blind source separation based on endpoint estimation with application to the mlsp 2006 data competition. *Neurocomput.*, 72(1-3):47–56, December 2008.

[MBJS96] S. Makeig, A.J. Bell, T.P. Jung, and T.J. Sejnowski. Independent component analysis of electroencephalographic data. *Advances in Neural Information Processing Systems*, 8:145–151, 1996.

[McC87] P. McCullagh. *Tensor Methods in Statistics*. Chapman and Hall, London, 1987.

[McK98] K. I. M. McKinnon. Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM J. on Optimization*, 9(1):148–158, May 1998.

[Mey11] G. Meyer. *Geometric optimization algorithms for linear regression on fixed-rank matrices*. PhD thesis, University of Liège, 2011.

[MFTM01] David R. Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Technical Report UCB/CSD-01-1133, EECS Department, University of California, Berkeley, Jan 2001.

[MuAWA10] Tahir Nadeem Malik, Azzam ul Asar, Mudasser F. Wyne, and Shakil Akhtar. A new hybrid approach for the solution of nonconvex economic dispatch problem with valve-point effects. *Electric Power Systems Research*, 80(9):1128 – 1136, 2010.

[NI08] Nasimul Noman and Hitoshi Iba. Differential evolution for economic load dispatch problems. *Electric Power Systems Research*, 78(8):1322 – 1331, 2008.

[Nik10] Taher Niknam. A new fuzzy adaptive hybrid particle swarm optimization algorithm for non-linear, non-smooth and non-convex economic dispatch problem. *Applied Energy*, 87(1):327 – 339, 2010.

[NM65] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[NN87] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Society for Industrial Mathematics, 1987.

[NW00] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[O'R85] Joseph O'Rourke. Finding minimal enclosing boxes. *Internat. J. Comput. Inform. Sci.*, 14:183–199, June 1985.

[Pha00] Dinh Tuan Pham. Joint approximate diagonalization of positive definite hermitian matrices. *SIAM J. Matrix Anal. Appl.*, 22(4):1136–1152, July 2000.

[PI96] Dinh Tuan Pham and Member Ieee. Blind separation of instantaneous mixture of sources based on order statistics. *IEEE Trans. Signal Processing*, 44:2768–2779, 1996.

[PLSL05] Jong-Bae Park, Ki-Song Lee, Joong-Rin Shin, and K.Y. Lee. A particle swarm optimization for economic dispatch with nonsmooth cost functions. *Power Systems, IEEE Transactions on*, 20(1):34 – 42, feb. 2005.

[PSL05] Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.

[PV06] Dinh-Tuan Pham and Frédéric Vrins. Discriminacy of the minimum range approach to the simultaneous blind separation of bounded sources. In M. Verleysen, editor, *Advances in Computational Intelligence and Learning*, pages 377–382, April 2006. 14th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

[RAD+10] Quentin Rentmeesters, P.-A. Absil, Paul Van Dooren, Kyle Gallivan, and Anuj Srivastava. An efficient particle filtering technique on the grassmann manifold. In *Proceedings of the 35th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2010)*, pages 3838–3841, 2010.

[Roc63] R.T. Rockafellar. *Convex functions and dual extremum problems*. PhD thesis, Harvard University, 1963.

[Roc03] R. T. Rockafellar. A property of piecewise smooth functions. *Comput. Optim. Appl.*, 25(1-3):247–250, March 2003.

[SAQ+12] S. E. Selvan, U. Amato, C. Qi, K. A. Gallivan, M. F. Carfora, M. Larobina, and B. Alfano. Descent algorithms on oblique manifold for source-adaptive ICA contrast. *IEEE Transactions on Neural Networks and Learning Systems*, 23(12):1930–1947, December 2012.

[Sar09] Alain Sarlette. *Geometry and Symmetries in Coordination Control*. PhD thesis, University of Liège, 2009.

[SCAA12] E.S. Suviseshamuthu, A. Chattopadhyay, U. Amato, and P.A. Absil. Range-based non-orthogonal ica using cross-entropy method. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2012)*, 2012.

[SCC03] N Sinha, R Chakrabarti, and RK Chattopadhyay. Evolutionary programming techniques for economic load dispatch. *IEEE Transactions on Evolutionary Computation*, 7(1):83–94, feb 2003.

[SDH09] H. Shen, K. Diepold, and K. Hüper. Geometric algorithms for the non-whitened one-unit linear independent component analysis problem. In *Proceedings of the* 15^{th} *IEEE Workshop on Statistical Signal Processing (SSP 2009)*, pages 133–138, Cardiff, Wales, UK, 2009.

[SEL03] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME – A flexible appearance modelling environment. *IEEE Transactions on Medical Imaging*, 22(10):1319–1331, October 2003.

[Sel11] A. Immanuel Selvakumar. Enhanced cross-entropy method for dynamic economic dispatch with valve-point effects. *International Journal of Electrical Power & Energy Systems*, 33(3):783 – 790, 2011.

[SH09] Hao Shen and Knut Huper. Block jacobi-type methods for non-orthogonal joint diagonalisation. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '09, pages 3285–3288, Washington, DC, USA, 2009. IEEE Computer Society.

[Sha70] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, July 1970.

[SHP+10] Nidul Sinha, Kaustabh Moni Hazarika, Shantanu Paul, Himanshu Shekhar, and Amrita Karmakar. Improved real quantum evolutionary algorithm for optimum economic load dispatch with non-convex loads. In Bijaya Panigrahi, Swagatam Das, Ponnuthurai Suganthan, and Subhransu Dash, editors, *Swarm, Evolutionary, and Memetic Computing*, volume 6466 of *Lecture Notes in Computer Science*, pages 689–700. Springer Berlin / Heidelberg, 2010.

[SK10] Hao Shen and Martin Kleinsteuber. Complex blind source separation via simultaneous strong uncorrelating transform. In *Proceedings of the 9th international conference on Latent variable analysis and signal separation*, LVA/ICA'10, pages 287–294, Berlin, Heidelberg, 2010. Springer-Verlag.

[SP97] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, December 1997.

[SRPP11] S. Seth, M. Rao, Il Park, and J. C. Principe. A unified framework for quadratic measures of independence. *Trans. Sig. Proc.*, 59(8):3624–3635, August 2011.

[Ste71] E.M. Stein. *Singular integrals and differentiability properties of functions (PMS-30)*, volume 30. Princeton university press, 1971.

[TL02] Nickolay T. Trendafilov and Ross A. Lippert. The multimode procrustes problem. *Linear Algebra and its Applications*, 349(1-3):245–264, July/Summer 2002.

[Tor97] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[Tou83] Godfried Toussaint. Solving geometric problems with the rotating calipers. In *In Proc. IEEE MELECON 1983*, pages 10–02, 1983.

[Tuc64] L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology*, pages 109–127. Holt, Rinehart & Winston, NY, 1964.

[Tuc66] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[vdBE02] F. van den Bergh and AP Engelbrecht. A new locally convergent particle swarm optimiser. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, volume 3, pages 6–pp. IEEE, 2002.

[vdV01] A.-J. van der Veen. Joint diagonalization via subspace fitting techniques. In *Proceedings of the Acoustics, Speech, and Signal Processing, 2001. on IEEE International Conference - Volume 05*, ICASSP '01, pages 2773–2776, Washington, DC, USA, 2001. IEEE Computer Society.

[VJ04] TAA Victoire and AE Jeyakumar. Hybrid PSO-SQP for economic dispatch with valve-point effect. *Electric Power Systems Research*, 71(1):51–59, SEP 2004.

[VJ05] TAA Victoire and AE Jeyakumar. Deterministically guided PSO for dynamic dispatch considering valve-point effect. *Electric Power Systems Research*, 73(3):313–322, MAR 2005.

[vLA87] P. J. van Laarhoven and E. H. Aarts. *Simulated Annealing: Theory and Applications (Mathematics and Its Applications)*. Springer, 1 edition, June 1987.

[VLV07] Frédéric Vrins, John Aldo Lee, and Michel Verleysen. A minimum-range approach to blind extraction of bounded sources. *IEEE Transactions on Neural Networks*, pages 809–822, 2007.

[Vri07] F. Vrins. *Contrast properties of entropic criteria for blind source separation: a unifying framework based on information-theoretic inequalities.* PhD thesis, Université catholique de Louvain, 2007.

[VT03] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *Proc. Computer Vision and Pattern Recognition Conf. (CVPR '03), Madison, WI*, volume 2, pages 93–99, 2003.

[VV09] A. I. F. Vaz and L. N. Vicente. Pswarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods Software*, 24(4-5):669–685, 2009.

[Wei08] Thomas Weise. Global optimization algorithms – theory and application –, 2008.

[Wol69] P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[Wri95] M. Wright. Direct search methods: Once scorned, now respectable, 1995.

[WS93] D.C. Walters and G.B. Sheble. Genetic algorithm solution of economic dispatch with valve point loading. *Power Systems, IEEE Transactions on*, 8(3):1325 –1332, aug 1993.

[Yer02] Arie Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Transactions on Signal Processing*, 50(7):1545–1553, 2002.

[YÖ11] Celal Yaşar and Serdar Özyön. A new hybrid approach for nonconvex economic dispatch problem with valve-point effect. *Energy*, 36(10):5838 – 5845, 2011.

[YWZY09] Xiaohui Yuan, Liang Wang, Yongchuan Zhang, and Yanbin Yuan. A hybrid differential evolution method for dynamic economic dispatch with valve-point effects. *Expert Systems with Applications*, 36(2, Part 2):4042 – 4048, 2009.

[YYH96] Hong-Tzer Yang, Pai-Chuan Yang, and Ching-Lien Huang. Evolutionary programming based economic dispatch for units with non-smooth fuel cost functions. *Power Systems, IEEE Transactions on*, 11(1):112 –118, feb 1996.

[ZZL05] Jun Zhang, Jing-Ru Zhang, and Kang Li. A sequential niching technique for particle swarm optimization. In *Advances in Intelligent Computing*, volume 3644 of *Lecture Notes in Computer Science*, pages 390–399. Springer-Verlag, 2005.