

Springer Series in Operations Research  
and Financial Engineering

Charles Audet  
Warren Hare

# Derivative-Free and Blackbox Optimization

# **Springer Series in Operations Research and Financial Engineering**

## **Series Editors**

Thomas V. Mikosch  
Sidney I. Resnick  
Stephen M. Robinson

More information about this series at <http://www.springer.com/series/3182>

Charles Audet • Warren Hare

# Derivative-Free and Blackbox Optimization



Springer

Charles Audet  
Dépt. Mathématiques et Génie Industriel  
Ecole Polytechnique de Montréal  
Montréal, Québec, Canada

Warren Hare  
Department of Mathematics  
University of British Columbia  
Kelowna, BC, Canada

ISSN 1431-8598                    ISSN 2197-1773 (electronic)  
Springer Series in Operations Research and Financial Engineering  
ISBN 978-3-319-68912-8        ISBN 978-3-319-68913-5 (eBook)  
<https://doi.org/10.1007/978-3-319-68913-5>

Library of Congress Control Number: 2017958449

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# *Foreword*

---

This timely book provides an introduction to blackbox optimization. This is often shortened to BBO, and it refers to an optimization problem in which some or all of the problem functions are gotten by running a computer subroutine with the optimization trial point in the input parameters. Some of these routines are very expensive to execute for a given argument. I was involved in a problem where the cost of a function value was three weeks on four SGI processors. Despite the expense, BBO is a topic of crucial importance to many applications. This is especially true for important design applications based on legacy simulation codes that do not lend themselves to automatic differentiation or the computation of adjoints.

I was told by an engineering executive that his company felt little incentive to commit scarce resources to redo these legacy codes to make it more likely for them to produce derivatives using adjoint solvers or automatic differentiation. His reasoning was since his department had not done nearly everything they could with the legacy codes he was not ready to pay for the reimplementation. This is unlikely to change anytime soon; research money is always tight. Thus, DFO algorithms, optimization algorithms that do not use derivatives to calculate steps, are unlikely to decrease in importance for applications. Besides, I have also been told by some engineers that they do not trust the optima found using gradient-based algorithms because of the belief, justified or not, that they are too dependent on the starting point of the gradient-based algorithm. There is some truth to this as you will see if you use a DFO algorithm that incorporates a more broadly based exploration of design space, like the MADS or surrogate-based algorithms covered in this book. Still, I do not want to overemphasize this point. I would lean towards using derivatives if I had them.

Another important aspect of BBO treated in the book is the various forms taken by the constraints in such problems. They may not be numerical in the sense that a given set of the constraints may just return a “yes” or a “no” telling whether the argument is feasible without a number quantifying the violation.

If you want your code to be used for more than academically staged beauty contests involving test problems with analytic or no constraints, then you had better be able to deal with nonnumerical constraints. These include the so-called hidden constraints for which infeasibility is indicated by not returning a meaningful value from the underlying simulation. I was involved in a project once for which the objective function value of 2.6 was often returned. It turned out that the programmer had decided to return 2.6 whenever the objective function subroutine failed to complete the function or constraint evaluation. Unfortunately, this was well within the range of meaningful function values and so we scratched our heads for an hour or so over some strange optimization steps. I was involved in another application for which hidden constraints were violated at about two-thirds of our trial points.

Let me be clear that artificial test problems may be difficult, and nonnumerical constraints may or may not be difficult to treat for a given problem, though obviously they do require different approaches than numerical constraints. My contention is that it is pointless to advertise a DFO routine as a BBO routine if it cannot treat nonnumerical constraints.

Efficiency is another issue that has a different flavor for BBO problems. It is nicer to take fewer expensive calls to the blackbox routine to get an optimum, but it is not all that counts. Reliability in finding an optimizer with not too many evaluations is also important. One industrial group decided to use MADS in their in-house software for an important application because it never took more than 15 minutes to find an answer for instances of a recurring problem. Their own excellent nonlinear interior point code sometimes took milliseconds to find an answer, but sometimes it failed. Fifteen minutes was acceptable, but failure was expensive because of equipment that had to wait until the problem was solved each time.

The hardest decision a textbook writer has to make is what to leave out. There are always topics trying to worm their way into the book. If you let them in, then you may find that you have written an encyclopedic volume that is expensive for the student to buy and a difficult book to teach from unless the teacher is an expert. Some optimization books are like this. I believe that Charles and Warren have made reasonable choices to avoid that trap.

Charles and Warren are an interesting team. Charles is a major research contributor to the subject with some truly amazing applications under his belt; my personal favorite is the placement of Québec snowfall sensors—lots of nonnumerical constraints.

Warren has come more lately to this area, and that gives him a fresher eye than Charles might have in knowing what topics are difficult for the novice and how much detail is needed to explain them. My spot reading in the book makes me think that this would be a fine text to teach from at the advanced undergraduate/beginning graduate level. It does not try to do too much or too little on any topic. I really enjoyed the lucid presentation in

Chap. 5 of the McKinnon example showing that pure Nelder-Mead can fail analytically—we all knew it could fail numerically. Of course, that does not stop its use, it just needs to be restarted with a fresh simplex when it fails. However, that is an inconvenience.

This book is explicit about the desire to make nonsmooth analysis a topic studied by all optimizers, and I hope it will accomplish this important goal. This book does a thorough job presenting the theory behind the methods treated here, and that requires nonsmooth analysis. I applaud the authors for including the theory as well as the practical details for the algorithms treated. It is more common to do one or the other.

It is always irritating to me when important researchers say, or imply, that theory is not important for methods intended to solve real problems. I could not disagree more. A theorem should say what happens when a method is applied to a problem when the hypotheses hold. It is a proof of correctness for the algorithm on the class of problems satisfying the hypotheses. It is not that the hypotheses should be checked before the algorithm is used. I will try to make my point with a simple example: the theory says that a square matrix is positive definite iff the Cholesky decomposition of that matrix exists. But, we do not check to see that the matrix is positive definite before attempting the decomposition. If the decomposition fails, then we know.

I firmly believe that if a method works well on a class of problems, then mathematics can give insight into why this happens. It is only that we are sometimes unable to discover the right mathematics. But we must keep trying. This book gives valuable background for the quest, which I hope will be taken up by those who study this book.

Noah Harding Professor Emeritus  
Department of Computational  
& Applied Mathematics  
Rice University  
Houston, TX, USA

John E. Dennis, Jr.

# *Preface*

---

This book is inspired by one goal and one belief. Our goal is to provide a clear grasp of the foundational concepts in derivative-free and blackbox optimization, in order to push these areas into the mainstream of nonlinear optimization. Our belief is that these foundational concepts have become sufficiently mature that it is now possible to teach them at a senior undergraduate level.

Derivative-free and blackbox optimization have made massive advances over the past two decades and, in our opinion, represent one of the most rapidly expanding fields of nonlinear optimization research. We also feel that derivative-free and blackbox optimization represent one of the most important areas in nonlinear optimization for solving future applications in real-world problems.

We target this book at two broad audiences and hope that both will find value.

Our first audience is individuals interested in entering (or better understanding) the fascinating world of derivative-free and blackbox optimization. We do not present the absolute state of the art in modern algorithms and theory, as we feel that it belongs in the realm of research papers. Instead we focus on the foundational material required to understand and appreciate the state of the art. To this end, in addition of studying several optimization methods, this book includes an elementary introduction to parts of nonsmooth analysis that have proved useful in conceiving and analyzing the methods given here. Nonsmooth analysis is not covered often enough outside the nonsmooth research community, and we hope our small contribution will help bridge this gap. The book also presents rigourous convergence theory for the algorithms in a way suitable for students in the mathematical sciences or in engineering. We hope that if more students see what can be done, both theoretically and practically, via derivative-free optimization, they may choose to make careers in this rewarding research area. Moreover, we hope this book will leave them well prepared to step into the world of derivative-free and blackbox optimization.

The second audience is practitioners who have real-world problems to solve that cannot be approached by traditional gradient-based methods. In the past, many such practitioners have fallen back on *ad hoc* methods, resulting in a plethora of papers publishing incremental improvements to solution quality. The methods covered in this book have proven convergence results, mathematically supported stopping criterion, and a track record of practical success. Yet, for all of this, the methods are nonetheless easy to use and elegant in their simplicity. While we do not feel that the methods herein are the only possible approaches to real-world problems, we hope they can provide better and more consistent starting points for future applications.

Finally, we remark that, for pedagogical purposes, we will present our algorithms not necessarily as they were discovered, but as hindsight makes us wish they had been.

**Some Special Thanks.** From Charles Audet’s point of view, this book began through joint work with John Dennis. Certain portions of this book come from notes generated during past projects, and we are very grateful for these early interactions. From Warren Hare’s point of view, this book began through the University of British Columbia supporting a sabbatical year, much of which was spent in the GERAD Research Center at the Université de Montréal Campus.

Both authors wish to express special thanks to Sébastien Le Digabel, who helped design the table of contents and only avoided further collaboration (on this project) by becoming a father for the second time. Thanks also to Catherine Poissant, who diligently ensured that all exercises were doable, and to Christophe Tribes and Yves Lucet, for helping with some of the numerical examples within this book. Finally, thanks to the students of the winter 2017 classes in Kelowna and in Montréal and to Chayne Planiden and Viviane Rochon Montplaisir, who proofread and made some precious and constructive comments on the book.

Special thanks to the Banff International Research Station (BIRS) for hosting us during the final stages of completing this book.

And, of course, a huge thank-you to our families – Annie, Xavier, Blanche, Émile, Shannon, and Alden – for supporting us through this project and understanding the importance of properly optimized octagonal whiskey glasses.

**Expected Background.** Readers of this textbook are expected to have an understanding of multivariate calculus (gradients, Hessians, multi-variable integration, etc.), linear algebra (vectors, matrices, determinants, linear independence, etc.), and mathematical proof techniques (contraposition, induction, contradiction, etc.). These subjects are typically taught in first or second year of mathematics, engineering, computer science, and other degrees focused in quantitative science. We recommend completing courses in each of these before reading this book.

**How to Use This Book.** This book is designed as a textbook, suitable for self-learning or for teaching an upper-year university course. We envision the book as suitable for either a one- or two-term course and have designed it to be modular, so it is not necessary to read cover to cover, from page 1 to page 300. Nonetheless, certain portions should be read before continuing through the book. The book is separated into five main parts, plus an appendix.

- **Part 1: Introduction and Background Material.** This part contains introductory material and the necessary background for reading the remainder of the book. Chapter 1 (Introduction: Tools and Challenges) introduces the ideas and purpose of derivative-free optimization (DFO) and blackbox optimization (BBO) and includes three motivational examples. Chapter 2 (Mathematical Background) provides definitions and notations used throughout the book, including brief primers on convexity and simplices. These two concepts are of great importance throughout the book, so we recommend that readers are comfortable with the material before moving further. Chapter 3 (The Beginnings of DFO Algorithms) introduces the first algorithms in the book. While the algorithms in Chapter 3 are naive and generally ineffective, they provide the building blocks for many other methods and their analyses. Therefore, while not strictly necessary, we recommend that readers study this chapter before moving further.
- **Part 2: Popular Heuristic Methods.** This part examines classical heuristics for solving optimization problems without using gradient evaluations. While these methods do not meet *our* definition of a DFO method, they are nonetheless popular and effective methods for solving optimization problems without using gradients. Chapter 4 (Genetic Algorithms) can be read as a stand-alone chapter, as can Chapter 5 (Nelder-Mead). Part 2 can be skipped entirely without affecting understanding of Part 3 or 4.
- **Part 3: Direct Search Methods.** This part focuses on the first of two major categories of DFO methods: direct search methods. In these methods, the DFO algorithm evaluates function values at a collection of points and acts based on those values without any derivative approximation. As such, the methods can be effective in BBO even when the objective and constraint functions are nonsmooth. These methods apply and expand on results from linear algebra and provide a great opportunity to delve into this subject. Chapter 6 (Positive Bases and Nonsmooth Optimization) provides some important background material for understanding direct search methods, including elements of the nonsmooth analysis. Chapter 7 (Generalised Pattern Search) explores a basic direct search method for unconstrained optimization and can be read after completing the first four sections of Chapter 6. Chapter 8 (Mesh

Adaptive Direct Search) advances from Chapter 7, so it should only be read after completing that chapter. The method proposed in Chapter 8 handles nonsmooth constraints and relies on the material from Section 6.5 for the convergence analysis. Part 3 can be skipped entirely without affecting understanding of Part 2 or 4.

- **Part 4: Model-Based Methods.** This part focuses on the second of two major categories of DFO methods: model-based methods. In these methods, the DFO algorithm uses function values to build an approximation model of the objective and uses the model to guide future iterations. These methods are effective when the objective function is expected to be smooth (although gradients are not analytically available). The techniques use numerical analysis to prove that classical gradient-based algorithms can be applied when gradients are approximated. Chapter 9 (Building Linear and Quadratic Models) is a stand-alone chapter that shows how to construct linear and quadratic models of the objective function. Chapter 9 is designed to answer a key question raised in Chapters 10 and 11. Chapters 10 (Generalised Model-Based Descent) and 11 (Model-Based Trust Region) are mostly stand-alone chapters that cover two popular frameworks for model-based DFO. However, both chapters use definitions from Section 9.1, so they should only be read after completing that section. Part 4 can be skipped entirely without affecting understanding of Part 2 or 3.
- **Part 5: Extensions and Refinements.** This part discusses how to develop and use BBO and DFO methods efficiently in practice. Chapter 12 (Variables and Constraints) discusses various types of variables and proposes a more subtle approach to handle relaxable constraints than the simple strategy of rejecting any infeasible point. Chapter 13 (Optimization Using Surrogates and Models) discusses a way to use surrogates and models of the objective function and constraints, which is crucial for improving efficiency. This chapter relies on Part 3 for the algorithmic mechanism and on Chapter 9 for building models. Chapter 14 (Biobjective Optimization) discusses situations where there is not one but two conflicting objective functions to be optimized. It shows how to approach this question by solving a series of single-objective reformulations. Each chapter of this part is independent and may be skipped.
- **Appendix: Comparing Optimization Methods.** This appendix provides a very brief overview of good practices when performing numerical tests to compare optimization algorithms. The part also includes a large project, which applies material from Parts 1, 2, 3, and 4 in a single problem.

In order to help the book be a suitable teaching aid, we have strived to make each chapter approximately equal to 3 hours of lecture and included a collection of exercises at the end of each chapter.

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ) and the more difficult ones by a plus symbol (+). Some are tagged by both symbols.

For readers interested in the history of the subject, each part concludes with **Remarks** that provide references and historical context. These only provide some very high-level remarks and are by no means a complete historical overview. The Remark portions also include some comments on new results in the field and modern research directions.

Montréal, QC, Canada  
Kelowna, BC, Canada

Charles Audet  
Warren Hare

# *Contents*

---

|   |    |
|---|----|
| Foreword  | V  |
| Preface   | IX |
| <b>Part 1. Introduction and Background Material</b>   | 1  |
| Chapter 1. Introduction: Tools and Challenges in Derivative-Free<br>and Blackbox Optimization | 3  |
| 1.1. What Are Derivative-Free and Blackbox Optimization?                                      | 3  |
| 1.2. Classifications of Optimization Problems   | 6  |
| 1.3. Example Applications   | 8  |
| 1.4. Remarks on Blackbox Optimization Problems  | 11 |
| Chapter 2. Mathematical Background  | 15 |
| 2.1. Vectors, Matrices, and Norms   | 16 |
| 2.2. Functions and Multivariate Calculus  | 17 |
| 2.3. Descent Directions   | 20 |
| 2.4. Basic Properties of Sets   | 21 |
| 2.5. Convexity  | 22 |
| 2.6. Simplices  | 24 |
| Chapter 3. The Beginnings of DFO Algorithms   | 33 |
| 3.1. Exhaustive Search  | 33 |
| 3.2. Grid Search  | 35 |
| 3.3. Coordinate Search  | 37 |
| 3.4. Selecting a Starting Point   | 40 |
| 3.5. Convergence Analysis of Coordinate Search  | 41 |
| 3.6. Algorithmic Variants of Coordinate Search  | 44 |
| 3.7. Methods in This Book   | 47 |
| Some Remarks on DFO   | 53 |

|   |     |
|---|-----|
| <b>Part 2. Popular Heuristic Methods</b>                                    | 55  |
| Chapter 4. Genetic Algorithms   | 57  |
| 4.1. Biology Overview and the GA Algorithm                                  | 58  |
| 4.2. Fitness and Selection  | 59  |
| 4.3. Encoding   | 62  |
| 4.4. Crossover and Mutation   | 64  |
| 4.5. Convergence and Stopping   | 67  |
| Chapter 5. Nelder-Mead  | 75  |
| 5.1. The NM Algorithm   | 76  |
| 5.2. The Nelder-Mead Simplex  | 78  |
| 5.3. Convergence Study  | 80  |
| 5.4. The McKinnon Example   | 81  |
| Further Remarks on Heuristics   | 90  |
| <b>Part 3. Direct Search Methods</b>  | 93  |
| Chapter 6. Positive Bases and Nonsmooth Optimization                        | 95  |
| 6.1. Positive Bases   | 96  |
| 6.2. Constructing Positive Bases  | 99  |
| 6.3. Positive Bases and Descent Directions                                  | 101 |
| 6.4. Optimality Conditions for Unconstrained Problems                       | 101 |
| 6.5. Optimality Conditions for Constrained Problems                         | 107 |
| Chapter 7. Generalised Pattern Search                                       | 115 |
| 7.1. The GPS Algorithm  | 116 |
| 7.2. Opportunistic Strategy, the search Step, and Starting Points Selection | 119 |
| 7.3. The Mesh   | 120 |
| 7.4. Convergence  | 122 |
| 7.5. Numerical Experiments with the Rheology Problem                        | 129 |
| Chapter 8. Mesh Adaptive Direct Search                                      | 135 |
| 8.1. The MADS Algorithm   | 136 |
| 8.2. Opportunistic Strategy, the search Step, and Starting Points Selection | 140 |
| 8.3. Convergence Analysis of MADS   | 140 |
| 8.4. Dense Sets of Polling Directions                                       | 143 |
| 8.5. Further Experiments with the Rheology Optimization Problem             | 148 |
| Further Remarks on Direct Search Methods                                    | 154 |

|  |     |
|--|-----|
| <b>Part 4. Model-Based Methods</b>                             | 157 |
| Chapter 9. Building Linear and Quadratic Models                | 159 |
| 9.1. Fully Linear Models                                       | 160 |
| 9.2. Linear Interpolation                                      | 163 |
| 9.3. Error Bounds for Linear Interpolation                     | 166 |
| 9.4. Linear Regression   | 170 |
| 9.5. Quadratic Models  | 172 |
| Chapter 10. Model-Based Descent                                | 183 |
| 10.1. Controllable Accuracy                                    | 184 |
| 10.2. The MBD Algorithm  | 185 |
| 10.3. Flexibility in the MBD Algorithm and Stopping Criteria   | 187 |
| 10.4. The MBD Algorithm Has Successful Iterations              | 188 |
| 10.5. Convergence  | 194 |
| 10.6. Additional Experiments with the Rheology Problem         | 196 |
| Chapter 11. Model-Based Trust Region                           | 201 |
| 11.1. The MBTR Algorithm                                       | 202 |
| 11.2. Model Checks and Stopping Conditions                     | 205 |
| 11.3. Solving the Trust Region Subproblem                      | 206 |
| 11.4. Convergence  | 209 |
| 11.5. More Experiments with the Rheology Optimization Problem  | 213 |
| Further Remarks on Model-Based Methods                         | 217 |
| <b>Part 5. Extensions and Refinements</b>                      | 219 |
| Chapter 12. Variables and Constraints                          | 221 |
| 12.1. Types of Variables                                       | 222 |
| 12.2. Types of Constraints                                     | 224 |
| 12.3. The Constraint Violation Function                        | 225 |
| 12.4. Relaxable Constraints by the Progressive Barrier         | 228 |
| Chapter 13. Optimization Using Surrogates and Models           | 235 |
| 13.1. Surrogate Problem and Surrogate Functions                | 235 |
| 13.2. The Surrogate Management Framework                       | 238 |
| 13.3. Final Experiments with the Rheology Optimization Problem | 241 |
| Chapter 14. Biobjective Optimization                           | 247 |
| 14.1. The Pareto Set and Front                                 | 248 |
| 14.2. Single-Objective Approaches                              | 249 |
| 14.3. Biobjective Optimization Algorithm                       | 253 |
| Final Remarks on DFO/BBO                                       | 261 |

|  |     |
|--|-----|
| Appendix A. Comparing Optimization Methods | 263 |
| A.1. Test Sets                             | 264 |
| A.2. Data Collection                       | 265 |
| A.3. Data Analysis                         | 267 |
| Solutions to Selected Exercises            | 281 |
| References                                 | 289 |
| Index                                      | 299 |

PART

# 1

## Introduction and Background Material

Part 1 introduces and contextualises the elements of the book.

- Chapter 1 introduces tools and challenges in derivative-free and in blackbox optimization. It also presents examples of applications that will be used throughout the book.
- Chapter 2 presents the mathematical background material necessary to read the book. It can be read on a need-to-know basis.
- Chapter 3 shows what not to do. Exhaustive and grid search methods consume an unacceptable number of function evaluations. A first elementary method, the coordinate search algorithm, generates comparable solutions with a small fraction of the function evaluations.

Conclusion: there is a need for more elaborate optimization methods.

# *Introduction: Tools and Challenges in Derivative-Free and Blackbox Optimization*

In this introductory chapter, we present a high-level description of optimization, blackbox optimization, and derivative-free optimization. We introduce some basic optimization notation used throughout this book, and some of the standard classifications of optimization problems. We end with three examples where blackbox optimization problems have arisen in practice. The first of these examples will be used throughout this book as a case study for how various optimization algorithms behave.

## **1.1. What Are Derivative-Free and Blackbox Optimization?**

Optimization is the study of the mathematical problem of minimising or maximising a function  $f$ , possibly subject to some constraints such as the variables  $x$  lie in the constraint set  $\Omega$ . Such problems arise naturally in almost

every area of modern research. In oceanography, one could seek to identify the best location for tsunami detection buoys. In civil engineering, one might seek to minimise the damage done to a building during an earthquake. In pediatric cardiology, doctors want to identify the shape of a surgical graft to delay as much as possible the next operation. Chances are, if something can be expressed as a mathematical equation, then at some point somebody will want to minimise it (or some portion of it).

This definition of optimization actually encompasses several highly related problems. We could seek the *minimum function value* of  $f$  over the constraint set  $\Omega$ , which we mathematically denote

$$\min_x \{f(x) : x \in \Omega\}.$$

If instead we are interested in finding the point (or set of points) that provide the minimum function value  $z$ , then we seek the *argument of the minimum*:

$$\operatorname{argmin}_x \{f(x) : x \in \Omega\} := \{x \in \Omega : f(x) = z\}.$$

Note that, the minimum function value could be:

- i.  $-\infty$ : such as

$$\min_x \{x_1 : x \in \mathbb{R}^3\}.$$

- ii. A well-defined real number: such as

$$\min_x \{\|x\|^2 : x \in \mathbb{R}^2, x_1 \in [-1, 2], x_2 \in [0, 3]\}.$$

- iii. Undefined: such as

$$\min_x \{f(x) : x \in \mathbb{R}, x^2 = -1\}.$$

In the undefined case, we also have the situation where the *infimum* (greatest lower bound) of the function values exists, but no point actually provides the solution. For example, if we consider  $f(x) = e^x$  minimised over  $x \in \mathbb{R}$ , then the infimum is 0, but no point evaluates to 0. For the purpose of this book, we shall assume all problems either have a minimum function value of  $-\infty$ , or have the minimum function value being a well-defined real number. This assumption is simply to make presentation clearer, and is by no means standard in optimization research.

When exploring the argument of the minimum, we again have several situations that can occur. In particular, the argmin set can be:

- i. Empty: such as

$$\operatorname{argmin}_x \{x_1 : x \in \mathbb{R}^3\} = \emptyset.$$

- ii. A singleton: such as

$$\operatorname{argmin}_x \{\|x\|^2 : x \in \mathbb{R}^2, x_1 \in [-1, 2], x_2 \in [0, 3]\} = \{[0, 0]^\top\}.$$

- iii. A set of points: such as

$$\operatorname{argmin}_x \{\sin(x) : x \in \mathbb{R}, x \in [0, 7]\} = \{0, \pi, 2\pi\}.$$

iv. An infinite set: such as

$$\begin{aligned}\operatorname{argmin}_x\{0 : x \in [0, 2]\} &= [0, 2] \text{ or} \\ \operatorname{argmin}_x\{\cos(x) : x \in \mathbb{R}\} &= \{(2k + 1)\pi : k \in \mathbb{Z}\}.\end{aligned}$$

Related to  $\min$  and  $\operatorname{argmin}$  are their counterparts  $\max$ , the maximum function value, and  $\operatorname{argmax}$ , the argument of the maximum. In fact,  $\min$  and  $\max$  are so tightly related that it suffices to only consider one of them.

**LEMMA 1.1** (Relating Minimisation and Maximisation Problems).

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and  $\Omega \subseteq \mathbb{R}^n$  be such that the minimal value of  $f$  on  $\Omega$  exists or is  $-\infty$ . Then

$$\min_x\{f(x) : x \in \Omega\} = -\max_x\{-f(x) : x \in \Omega\}$$

and

$$\operatorname{argmin}_x\{f(x) : x \in \Omega\} = \operatorname{argmax}_x\{-f(x) : x \in \Omega\}.$$

**PROOF.** Exercise 1.1. □

In this book we focus exclusively on minimisation problems. Any need to maximise a function can be easily achieved via Lemma 1.1, or by making obvious adaptations to the algorithms in this book.

Now that we have a grasp on what we mean by optimization, we can define derivative-free optimization. Simply put,

Derivative-free optimization (DFO) is the mathematical study of optimization algorithms that do not use derivatives.

This statement has some subtleties that deserve exploring. Most notably, the definition states that the algorithms do not use derivatives, which is very different from saying that the objective function is nondifferentiable. The second subtlety is the term “mathematical study”, we shall discuss that phrase shortly.

One of the most common uses of DFO is in minimising an objective function that results from a computer simulation. The computer simulation could produce a fully differentiable function, but the output does not include derivatives. In this situation, gradient-based methods, such as Newton’s method, cannot be directly applied.

The term mathematical study helps to distinguish DFO from the larger field of blackbox optimization (BBO). In optimization, a blackbox is any process that when provided an input, returns an output, but the inner workings of the process are not analytically available. The most common form of blackbox is computer simulation, but other forms exist, such as laboratory experiments for example.

Blackbox optimization (BBO) is the study of design and analysis of algorithms that assume the objective and/or constraint functions are given by blackboxes.

In BBO, there are typically no assumptions of any form of continuity, differentiability or smoothness on the output.

BBO differs from DFO in two manners. First, for some optimization problems derivatives might exist, but obtaining or estimating them might be computationally expensive. Such problems could preferably be approached by derivative-free rather than blackbox methods. Second, BBO also includes research into heuristic methods or specialised *ad hoc* methods, for solving problems. Heuristic methods are designed to be rapid, but are generally unsupported by any rigorous or meaningful convergence analysis. As a mathematical study, DFO focuses on methods that can be mathematically analyzed to prove convergence and/or provide stopping criterion. This difference should become more clear as we proceed through this book. In fact, Part 2 of this book focuses on two popular heuristic methods for BBO. While we will provide some convergence theorems regarding the methods, the theorems will lack the strength of the results presented in Parts 3 and 4. This will help emphasise the difference between DFO and heuristic methods for blackbox functions.

On a final note about DFO and BBO, we make the strong and resounding comment to the reader.

If (generalised) gradient information is available, reliable, and obtainable at reasonable cost, then DFO and BBO will almost never outperform modern gradient-based methods. As such, if gradient information is available, reliable, and obtainable at reasonable cost, then gradient-based methods should be used.

While fairly obvious, it is somehow very easy to forget this comment. In summary, if derivatives are available, do not use DFO/BBO. If some derivative information exists, but requires a lot of effort to estimate, then DFO may be a good approach.

## 1.2. Classifications of Optimization Problems

In this section we recall some of the basic categories that optimization problems fall under.

The first major classification an optimization problem falls under is *continuous* or *discrete*. Continuous optimization problems are those where the *constraint space* allows a continuous selection of variables. This should not be confused with the objective function being continuous, which is classified later. In discrete problems, the constraint space does not allow a continuous selection of variables. For example:

- i.  $\min_x \{f(x) : x \in \mathbb{R}^n\}$  is a continuous optimization problem (regardless of  $f$ ).
- ii.  $\min_x \{\|x\|^2 : x_i \in \{-1, 0, 1\}\}$  is a discrete optimization problem (despite the objective being represented by a continuous function).
- iii.  $\min_x \{\|x\|^2 : x_1 \in [0, 1], x_2 \in \{0, 1\}\}$  is a discrete optimization problem, as the constraint space for  $x_2$  does not contain an interval of strictly positive length.
- iv.  $\min_x \{f(x) : x \in [0, 1] \cup [2, 3]\}$  is a continuous optimization problem, as the domain contains an interval of strictly positive length.

Most DFO algorithms focus on working with a continuous optimization problem. The purpose for this is twofold. First, assuming a continuous constraint space means that the concept of a limit can be invoked when examining the iterates of an algorithm. Second, most DFO methods perform some form of line search or shrink step. Without a continuous constraint space, it becomes difficult to work with this style of algorithm, as searching half way between two good solutions may suddenly be infeasible. While DFO algorithms for discrete optimization problems exist, this book focuses on the foundations of DFO, which assumes continuous optimization problems. Indeed, the only algorithms in this book that are easily adapted to discrete problems are the Genetic Algorithm (Chapter 4) and MADS (Chapter 8) algorithms. Other algorithms can be adapted, but the process is less obvious. Additional comments on this issue are presented in Chapter 12.

The next major classification of continuous optimization problems is *convex* versus *nonconvex*. Unlike continuous and discrete, this classification examines both the objective function and constraint set. If both the objective function and constraint set are *convex*, then the problem is deemed convex. Otherwise it is deemed nonconvex. As convexity is an important topic in optimization, we devote a portion of Chapter 2 to the subject.

Our final classification of continuous optimization problems is *smooth* versus *nonsmooth*. A smooth optimization problem is one where the objective function and constraint set are represented using continuously differentiable functions. Nonsmooth implies that there are at least some points where the objective function, or one of the constraint functions delimiting the feasible region does not have a well-behaved gradient. It should be noted that the classification of smooth versus nonsmooth can be quite tricky. Exercise 1.4 gives a classic example of a nonsmooth problem that can be reformulated as a smooth problem. Recall that DFO is the design and study of optimization algorithms where gradients are not used, which does not necessarily imply that the optimization problem is nonsmooth. Indeed, Section 1.3.2 provides an example where the objective function is smooth, but DFO is the logical solution approach.

Finally, we note that there are many further classifications of optimization problems: linear versus nonlinear, quadratic versus nonquadratic, semi-definite problems, least squares problems, etc. Each of these classifications

is important, and in general the more specific a classification a problem falls under, the more likely there are highly specialised algorithms designed for the problem.

### 1.3. Example Applications

In this section we present three examples of successful uses of DFO to solve real-world problems. Many other examples exist, some of which are mentioned in the end notes for Part 1 (page 53), and one of which is a parameter tuning example used in the project of Chapter 3 (page 51).

#### 1.3.1. Example Application 1: Positioning Tsunami-Detection

**Buoys.** After the Indian Ocean tsunami of December 2004 that killed 250,000 people in 14 countries, it was decided to place tsunami detection buoys in the Pacific Ocean. Each buoy is connected to its own vibration sensor lying on the ocean floor by a 6 km chain.

A tsunami is caused by underwater volcanic or seismic activities. Vibrations are detected by the sensor and relayed to the buoy that communicates with a satellite. The satellite then sends the information to the control centre. Information from all sensors is then processed, and extensive calculation using oceanographic information such as currents, winds, and topography are used to numerically estimate how the wave will propagate. In addition, there are constraints that require the buoys be located in a region in which the ocean floor is not too steep. They must be positioned near the tectonic fault for early detection, but not too close. Furthermore, if an underwater mountain lies between the buoy and the tsunami source, then the vibrations will be damped and incorrect predictions will follow.

Sophisticated simulation software was used to construct a blackbox function that takes as input the longitudinal and latitudinal position of each buoy, and returns as output the expected value of the warning time that the buoy deployment would give before a random (but realistic) tsunami hits a major coastal city. The optimization problem consists in finding the buoy placements that maximise the expected warning time. Optimizing the placement of 6 buoys resulted in problems with 12 variables, and each simulation required approximately 30 seconds to complete.

The complexity of the numerical simulation makes it unlikely that the warning time function is smooth, so using function evaluations to estimate derivatives is unlikely to be helpful. This makes direct search methods for BBO (see Part 3) a natural choice for this problem. Several variants of this problem have been considered using the MADS algorithm, and the results have been positive.

#### 1.3.2. Example Application 2: Improving Earthquake Retrofits.

Around the world, there are a number of very dense urban centres built on fault-lines. Chile, Indonesia, Japan, and Peru have all been struck by massive earthquakes since 2000, with damages reaching into billions of dollars each time. Damage, and risk to life, is particularly high in dense urban centres.

One reason is that such centres generally include very tall buildings that are in close proximity. During an earthquake, these buildings can “pound” into each other causing extra damage and increasing risk to structural instability. This is called the *pounding effect*, and has been documented and studied since at least the 1970s.

A number of engineering solutions have been proposed to reduce the pounding effect. One of the most common is to use *dampers* to join two adjacent buildings together. This reduces the chance of building colliding during earthquakes and has the added advantage of increasing structural stability on both buildings. As the upfront capital cost of installing dampers is high, some researchers have used mathematical models and optimization to minimise the number of dampers required, and maximise the effectiveness of an earthquake retrofit. Typical problems have 3 to 10 dampers, which results in optimization problems with 6 to 20 variables.

This problem presents some interesting features that make model-based DFO a natural choice. The objective function is a numerical integration generated through a blackbox earthquake simulator, and requires approximately 2 seconds per evaluation. As such, we do not have direct access to gradients, but as the objective function is constructed through integration, we have strong reasons to believe that it is differentiable. This suggests that a model-based DFO method (see Part 4) might be an appropriate approach.

Earthquake retrofitting has been approached by a variety of heuristic methods (including Genetic Algorithms and Nelder-Mead), as well as model-based DFO methods. The model-based DFO methods have consistently come out ahead.

**1.3.3. Example Application 3: Parameter Fit.** Consider the situation in which one wishes to build a mathematical model of a physical situation. The model depends on a set parameters whose values are unknown. How can optimization be used to find adequate values of these parameters?

We illustrate this on a simple example in the context of rheology of polymeric systems. This example will be used throughout the book to study algorithmic behaviours. Rheology is a branch of mechanics that studies properties of materials that determine their responses to mechanical force. The viscosity  $\eta$  of a material is often modeled as a function of the strain rate  $\dot{\gamma}$ :

$$\eta(\dot{\gamma}) = \eta_0(1 + \lambda^2\dot{\gamma}^2)^{\frac{\beta-1}{2}}.$$

This model is parameterised by the real numbers  $\eta_0$ ,  $\lambda$ , and  $\beta$ . In order to estimate their values, one could obtain data by performing laboratory experiments that would expose the material to various strain rates, and compile the observed viscosity. Table 1.1 displays observed values of the strain rate and the viscosity in a polymeric system.

TABLE 1.1. Observed data in rheology of polymeric systems

| Observation<br><i>i</i> | Strain rate<br>$\dot{\gamma}_i (s^{-1})$ | Viscosity<br>$\eta_i (Pa \cdot s)$ |
|-------------------------|--|------------------------------------|
| 1                       | 0.0137                                   | 3220                               |
| 2                       | 0.0274                                   | 2190                               |
| 3                       | 0.0434                                   | 1640                               |
| 4                       | 0.0866                                   | 1050                               |
| 5                       | 0.137                                    | 766                                |
| 6                       | 0.274                                    | 490                                |
| 7                       | 0.434                                    | 348                                |
| 8                       | 0.866                                    | 223                                |
| 9                       | 1.37                                     | 163                                |
| 10                      | 2.74                                     | 104                                |
| 11                      | 4.34                                     | 76.7                               |
| 12                      | 5.46                                     | 68.1                               |
| 13                      | 6.88                                     | 58.2                               |

Since these observations are empirical and contain measurement errors, there are no values of the parameters  $\eta_0$ ,  $\lambda$ , and  $\beta$  for which the model perfectly fits the data. We would like to identify the “best” values of these parameters. That is, we seek a parameter selection  $[\eta_0, \lambda, \beta]^\top$  such that  $\eta_0(1 + \lambda^2(\dot{\gamma}_i)^2)^{\frac{\beta-1}{2}} \approx \eta_i$  for each  $i \in \{1, 2, \dots, 13\}$ . Define the absolute error for data point  $i \in \{1, 2, \dots, 13\}$  by

$$\epsilon_i(\eta_0, \lambda, \beta) := \left| \eta_0(1 + \lambda^2(\dot{\gamma}_i)^2)^{\frac{\beta-1}{2}} - \eta_i \right|.$$

From here, we could seek to minimise the sum of the absolute errors, resulting in the nonsmooth optimization problem of minimising

$$(1.1) \quad \hat{g}(\eta_0, \lambda, \beta) = \sum_{i=1}^{13} \epsilon_i(\eta_0, \lambda, \beta).$$

Alternatively, we could seek to minimise the sum of the squared errors, resulting in the smooth optimization problem of minimising

$$(1.2) \quad \check{g}(\eta_0, \lambda, \beta) = \sum_{i=1}^{13} (\epsilon_i(\eta_0, \lambda, \beta))^2.$$

In either case, we have an unconstrained optimization problem, because the objective function allows any point  $(\eta_0, \lambda, \beta)$  to be considered. These two problems are used throughout the book to illustrate the behaviour of different optimization methods. However, it is worth noting that (generalised) gradient information is available for these problems, so DFO is not the right method of choice.

Parameter fitting problems define a broader class of optimization problem, and in many cases, DFO is the appropriate choice for parameter fitting problems. When the model function is well understood, then finding the optimal parameters can often be accomplished through (generalised) linear regression. However, if the model is not a simple analytical expression, then the problem becomes more difficult. For example, there are situations where the model consists of a blackbox computer code. In these situations, parameter fitting requires the use of DFO or BBO methods.

Parameter fitting using DFO has been successful applied in healthcare modelling, catalytic combustion kinetics, and web server queueing models. Some references can be found in the Part 3.7 remarks on DFO (page 53).

#### 1.4. Remarks on Blackbox Optimization Problems

Many other applications of BBO exist. They are typically small scale, having fewer than 50 variables, but each function evaluation can take from seconds to days. In general, applications occur in the following situations:

- i. The functions defining the problem are provided through a computer simulation that cannot be easily subjected to automatic differentiation.
- ii. The optimization problem involves conducting a laboratory experiment, so there are no explicit mathematical expressions.
- iii. The objective function is noisy, and gradient information is highly unreliable.
- iv. Although gradients are available, the practitioner chooses to use a BBO method.

As we remarked on page 6, this final usage is usually not appropriate. However, there are situations where a BBO method can find the solution before a gradient-based method can be implemented.

In this book, we focus on presenting the foundational material of DFO and BBO. As such, we will occasionally make simplifying assumptions, such as the objective function is continuous or differentiable. Such assumptions may often be relaxed without too much difficulty (although details become more technical). However, in order to present formal analyses, we make the implicit simplifying assumption that the objective and constraint functions are well behaved. This may seem benign, but in real-world applications, BBO often encounters difficulty related to a poorly behaved blackbox. Some examples that have arisen in real-world BBO applications include the following:

- v. Evaluation of the objective or constraint functions involves a Monte-Carlo simulation, and consequently a second evaluation at the same point may return a different value.
- vi. Due to bugs in a computer simulation, the objective function fails to return a meaningful value despite the point being feasible with respect to the known constraints. In the best case the simulation returns an error message, in the worst case the simulation stalls or crashes.

- vii. The objective function automatically rounds the output to a fixed precision, thus making numerical differentiation meaningless.

Dealing with such problems is one of the focuses of modern DFO and BBO research.

Another recurring theme in BBO/DFO is that the evaluation of the objective and constraint functions is often very computationally expensive. Thus, most research in this areas focuses on getting the most possible information out of each evaluation. With that in mind, we reiterate: if accurate gradients are available at reasonable cost, then a gradient-based method should be employed.

In some situations, a surrogate optimization problem is available, and is considerably cheaper to evaluate, but less accurate. One easy example is the case where the objective function launches a Monte-Carlo simulation. A surrogate problem can be created by running the simulation for a shorter period of time. Another example would be when physical factors such as friction are neglected in a ballistic model.

A surrogate can be very useful in helping ensure that an optimization algorithm gets the most possible information out of each evaluation. Developing good methods of building and using surrogates in BBO/DFO is another focus of modern research. Chapter 13 provides more information on surrogates, for now we conclude with a similar remark to our comment on gradients: if a reasonable surrogate model is readily available, then it should not be ignored.

## EXERCISES

---

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 1.1.** + Prove Lemma 1.1.

[Hint: be careful to consider all possible situations for the outcome of the problem.]

**EXERCISE 1.2.** For each optimization problem below, state whether the problem is *continuous* or *discrete*. If the problem is continuous, then state whether the problem is *smooth* or *nonsmooth*.

- $\min_{x \in \mathbb{R}^2} \{7x_1 + 14x_2 : x_1 \geq 0, x_2 \geq x_1\}.$
- $\min_{x \in \mathbb{R}} \{\sin(x_1) : x_1 \in \{0, 1, 2, 3, \dots\}\}.$
- $\max_{x \in \mathbb{R}^2} \{\text{floor}(x_1) + \text{ceiling}(x_2) : (x_1)^2 + (x_2)^2 \leq 1\}.$
- $\min_{x \in \mathbb{R}} \{x_1^2 : \cos(x_1) \geq 1\}.$

**EXERCISE 1.3.** Consider the pair of optimization problems

$$(P) \quad \min_{x \in \mathbb{R}^n} \{f(x) : x \in \Omega\} \quad \text{and} \quad (Q) \quad \min_{x \in \mathbb{R}^n, t \in \mathbb{R}} \{t : t \geq f(x), x \in \Omega\}.$$

- Show that if  $x \in \text{argmin}(P)$ , then  $[x, f(x)]^\top$  belongs to  $\text{argmin}(Q)$ .

- b) Show that if  $[x, t]^\top \in \operatorname{argmin}(Q)$ , then  $x$  belongs to  $\operatorname{argmin}(P)$ .

**EXERCISE 1.4.** Consider the following optimization problem

$$\min_x \left\{ \sum_{i=1}^n |x_i| : x \in \mathbb{R}^n, c(x) \leq 0 \right\},$$

where  $c(x)$  is a continuously differentiable function. As written, this problem is nonsmooth, as the objective function is not differentiable at 0 (see Definition 2.1). In this exercise, we construct an equivalent smooth problem.

- a) Let  $f(x) = \sum_{i=1}^n |x_i|$ . Rewrite the problem as the minimisation of the scalar  $\alpha \in \mathbb{R}$  over the variables  $[x, \alpha]^\top \in \mathbb{R}^n \times \mathbb{R}$  subject to the new constraint  $\alpha \geq f(x)$ . Why is this the same problem?
- b) Let  $\beta \in \mathbb{R}^n$  be a vector of variables. Prove that the single (non-smooth) constraint  $\beta_i \geq |x_i|$  can be written as two (smooth) constraints  $\beta_i \geq x_i$  and  $\beta_i \geq -x_i$ .
- c) Prove that the problem can be reformulated as a smooth problem with  $2n$  variables.

**EXERCISE 1.5.** + Consider the continuous nonsmooth optimization problem

$$\min_x \{F(x) : x \in \mathbb{R}^n, G(x) \leq 0\}$$

where  $F(x) = \max\{f_i(x) : i = 1, 2, \dots, N_F\}$ ,  $G(x) = \max\{g_i(x) : i = 1, 2, \dots, N_G\}$ , and where  $f_i$  and  $g_i$  are continuously differentiable functions or all  $i$ . Write an equivalent smooth optimization problem.

[Hint: see Exercise 1.4.]

**EXERCISE 1.6.** For each optimization problem below, state whether the problem is *continuous* or *discrete*. If the problem is continuous, then state whether the problem is *smooth* or *nonsmooth*.

- a)  $\max_x \{f(x) : x \in [\ell, u]\}$  where  $f$  is continuously differentiable,  $\ell \in \mathbb{R}^n$ ,  $u \in \mathbb{R}^n$ , and  $x \in [\ell, u]$  is interpreted as  $\ell_i \leq x_i \leq u_i$  for each  $i = 1, 2, \dots, n$ .
- b)  $\max_x \{F(x) : x \in C\}$  where  $F(x) = \max\{f_i(x) : i = 1, 2, \dots, N\}$ , and where  $f_i$  are continuously differentiable functions or all  $i$ , and  $C = \{x : \|x\| \geq 1\}$ .
- c)  $\min_t \left\{ \sum_{i=1}^n t_i : t_i(t_i - 1) = 0 \text{ for each } i = 1, 2, \dots, N \right\}.$

**EXERCISE 1.7.** A function  $f$  takes as input a vector  $p \in \mathbb{R}^n$  and uses it to build a model. The theoretical error of the model is represented by a definite integral and is estimated by a finite sum, which takes a few seconds to compute. The output of the function  $f$  is the value of the sum.

Consider the problem of finding values of  $p$  that minimise the estimation of the error. Explain why approaching this problem using DFO tools is appropriate.

**EXERCISE 1.8.** An engineer wishes to adjust the parameters of a machine so that the wear of one of the components is minimised. In order to measure the wear, the machine needs to run for 10 minutes using the prescribed parameters. Afterwards, the machine is stopped and the component is measured with a micrometer.

Consider the problem of finding values of  $p$  that minimise the wear. Explain why approaching this problem using BBO tools is appropriate.

**EXERCISE 1.9.**  $\square$  Using the programming language of your choice, code the functions  $\hat{g} : \mathbb{R}^3 \mapsto \mathbb{R}$  and  $\check{g} : \mathbb{R}^3 \mapsto \mathbb{R}$ , from equations (1.1) and (1.2) of the rheology optimization problem.

- Check that  $\hat{g}(5200, 140, 0.38) \approx 462.446$ .  
Compute  $\hat{g}(5252.0, 141.4, 0.3838)$ .
  - Check that  $\check{g}(5200, 140, 0.38) \approx 21,000.463$ .  
Compute  $\check{g}(5252.0, 141.4, 0.3838)$ .
  - Plot the difference between observed and predicted values as a function of  $\dot{\gamma}$  using  $\eta_0 = 5200$ ,  $\lambda = 140$  and  $\beta = 0.38$ , for both  $\hat{g}$  and  $\check{g}$ . A logarithmic scale helps readability.
- 



**Chapter 1 Project: Research an Applied BBO/DFO Problem.**  
Find an example of BBO or DFO arising in a real-world application. Examine the problem formulation and answer the following.

- Is the problem continuous or discrete?
- Is the problem smooth or nonsmooth (or impossible to know)?
- Why was BBO/DFO used in this case?
- Was it appropriate to use BBO/DFO in this case (why or why not)?

## *Mathematical Background*

For a full appreciation of the material in this book, it is assumed that the reader has followed courses in Multivariate Calculus and Linear Algebra. This chapter contains definitions, notation, and results that, although fairly common in mathematics literature, are not necessarily covered in standard multivariate calculus and linear algebra courses. Not all of this material will be used immediately and not all of the material is required for every chapter; however, all of this material shows up repeatedly throughout this book. We begin with an important comment on notation.

As this book is heavily invested in algorithms, in many places, vectors or parameters may depend on an iteration counter  $k$ . Note that superscripts will typically refer to the iteration number, or enumerate elements of a set. When raising a term to a power, if any chance of ambiguity occurs, then we will use parentheses. For example,  $\tau^2$  would mean the second value of  $\tau^k$ , while  $(\tau)^2$  would mean the value  $\tau$  squared.

It is safe to skip or skim-read the remainder of this chapter. However, be ready to return to it if necessary while proceeding through other portions of this book.

## 2.1. Vectors, Matrices, and Norms

In general,  $\mathbb{R}^n$  should be thought of as the space of  $n$  dimensional vectors, which we represent as column matrices

$$x \in \mathbb{R}^n \Leftrightarrow x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

For a matrix  $A \in \mathbb{R}^{m \times n}$  (or vector  $x \in \mathbb{R}^n$ ), we denote the *transpose* by  $A^\top$ , (or  $x^\top$  respectively).

The *norm* of a vector will be denoted  $\|x\|$  and unless otherwise stated should be taken to be the  $\ell_2$  norm (i.e., the Euclidean norm),

$$\|x\| = \|x\|_2 := \sqrt{(x_1)^2 + (x_2)^2 + \dots + (x_n)^2}.$$

We will also make use of the  $\ell_1$  norm and the  $\ell_\infty$  norm, denoted and defined via

$$\|x\|_1 := |x_1| + |x_2| + \dots + |x_n| \quad \text{and} \quad \|x\|_\infty := \max\{|x_1|, |x_2|, \dots, |x_n|\}.$$

Given a matrix  $A$ , we shall use the *induced matrix norm*,

$$\|A\|_* := \max\{\|Ax\|_* : \|x\|_* = 1\},$$

where  $*$  could denote the  $\ell_2$ ,  $\ell_1$ , or  $\ell_\infty$  norm. We shall also use the *Frobenius norm*, defined via

$$\|A\|_F = \left( \sum_{i=1}^m \sum_{j=1}^n A_{i,j}^2 \right)^{\frac{1}{2}} \geq \|A\|_2.$$

If the norm is not specified, it should be assumed that the vector norm is the  $\ell_2$  norm, and the matrix norm is the  $\ell_2$  induced matrix norm.

If  $A$  is a square matrix, then we denote the *determinant* of  $A$  by  $\det(A)$ . If  $\det(A) \neq 0$ , then  $A$  is *invertible* (also called nonsingular) and we denote the *inverse* by  $A^{-1}$ .

Finally, we recall that a set of vectors  $\{v^1, v^2, \dots, v^m\}$  is *linearly independent* if the only solution to

$$0 = \sum_{i=1}^m \lambda_i v^i \quad \lambda_i \in \mathbb{R}$$

is  $\lambda_i = 0$  for all  $i$ . The *span* of a set  $\Omega$  is denoted,  $\text{span}(\Omega)$ , and defined

$$\text{span}(\Omega) = \left\{ \sum_{i=1}^k \lambda_i x^i : k \geq 1, \lambda \in \mathbb{R}^k, x^i \in \Omega \right\}.$$

If a set of vectors  $V = \{v^1, v^2, \dots, v^m\} \subset \mathbb{R}^n$  is linearly independent and  $\text{span}(V) = \mathbb{R}^n$ , then we call that set of vectors a *basis* of  $\mathbb{R}^n$ . One of the key properties of a basis is that any basis of  $\mathbb{R}^n$  contains exactly  $n$  vectors.

## 2.2. Functions and Multivariate Calculus

We have already used the notation  $f : \mathbb{R}^n \mapsto \mathbb{R}$  to indicate that  $f$  is a function whose inputs are in  $\mathbb{R}^n$  and whose output is in  $\mathbb{R}$ . The notation  $f \in \mathcal{C}^0$  signifies that  $f$  is *continuous* and  $f \in \mathcal{C}^k$  signifies that  $f$  is  $k$  times *differentiable* and all *partial derivatives* are continuous up to the  $k$ -th order. For the sake of completeness, we recall the formal definition of differentiability.

**DEFINITION 2.1** (Differentiability and Gradient).

The function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is said to be *differentiable* at  $x \in \mathbb{R}^n$  if and only if there exists a vector  $g$  in  $\mathbb{R}^n$  such that

$$\lim_{u \rightarrow x} \frac{f(u) - f(x) - g^\top(u - x)}{\|u - x\|} = 0.$$

If  $f$  is differentiable at  $x$ , then there is a unique such vector  $g$ , and it is called the *gradient* of  $f$  at  $x$ , denoted  $\nabla f(x)$ .

If  $f$  is differentiable at some point  $x \in \mathbb{R}^n$ , then the gradient is the vector of  $\mathbb{R}^n$  composed of the  $n$  partial derivatives of  $f$ ,

$$\nabla f(x) = \left[ \frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]^\top.$$

The *directional derivative* of  $f$  at  $x$  in the direction of  $d \in \mathbb{R}^n$  is defined as

$$f'(x; d) := \lim_{t \searrow 0} \frac{f(x + td) - f(x)}{t}.$$

If the  $i$ -th partial derivative exists, it equals the directional derivative in the  $i$ -th coordinate direction:

$$\frac{\partial f(x)}{\partial x_i} = f'(x; e_i),$$

where  $e_i$  is the unit vector in the direction of the  $i$ -th coordinate (i.e., the vectors with a 1 in the  $i$ -th position and 0 in every other position). If  $f \in \mathcal{C}^1$ , then the directional derivatives satisfy

$$f'(x; d) = d^\top \nabla f(x) \quad \text{for every direction } d \in \mathbb{R}^n.$$

Using the fact that the scalar product of two vectors is equal to the product of their norms times the cosine of the angle that separates them, it follows that if  $f \in \mathcal{C}^1$  and  $x, d \in \mathbb{R}^n$  with  $\|d\| = 1$ , then  $|f'(x; d)| \leq \|\nabla f(x)\|$ , with equality if  $d$  and  $\nabla f(x)$  are colinear.

Observe that all directional derivatives of a function may exist, even if the function is not differentiable. For example, the directional derivative in the direction  $d \in \mathbb{R}^n$  of the function  $f(x) = \|x\|$  at 0 is

$$f'(0; d) = \lim_{t \searrow 0} \frac{f(0 + td) - f(0)}{t} = \lim_{t \searrow 0} \frac{\|td\|}{t} = \|d\|$$

but the gradient of  $f$  at 0 is undefined, since the limit in Definition 2.1 does not exist. The notion of directional derivative is more general than that of the gradient.

We next introduce the notion of *Lipschitz continuity*, which is stronger than continuous.

**DEFINITION 2.2 (Lipschitz Continuous).**

The function  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is said to be Lipschitz continuous on the set  $X \subseteq \mathbb{R}^n$  if and only if there exists a scalar  $K > 0$  for which

$$\|g(x) - g(y)\| \leq K\|x - y\| \quad \text{for all } x, y \in X.$$

The scalar  $K$  is called the Lipschitz constant of  $g$  relative to the set  $X$ .

Furthermore,  $g$  is said to be locally Lipschitz at  $\bar{x} \in \mathbb{R}^n$  (or Lipschitz near  $\bar{x} \in \mathbb{R}^n$ ) if  $g$  is Lipschitz on some open ball centred at  $x$ . That is, there exists  $K > 0$  and  $r > 0$  such that

$$\|g(x) - g(y)\| \leq K\|x - y\| \quad \text{for all } x, y \in B_r(\bar{x}) := \{z : \|z - \bar{x}\| < r\}.$$

The notation  $f \in \mathcal{C}^{0+}$  signifies that  $f$  is locally Lipschitz at all  $x \in \mathbb{R}^n$ .  $f \in \mathcal{C}^{0+}$  with constant  $K$  (near  $x$ ) means that  $f$  is Lipschitz continuous with Lipschitz constant  $K$  (on some open neighbourhood of  $x$ ). We expand this to  $f \in \mathcal{C}^{1+}$  with constant  $K$  to mean  $f \in \mathcal{C}^1$ , and the gradient,  $\nabla f$ , is Lipschitz continuous with Lipschitz constant  $K$ . Note that the  $K$  appearing in “ $f \in \mathcal{C}^{1+}$  with constant  $K$ ” refers to the Lipschitz constant of  $\nabla f$  and not of  $f$ ; i.e.,

$$\begin{aligned} & f \in \mathcal{C}^{1+} \text{ with constant } K \\ \Leftrightarrow & f \in \mathcal{C}^1 \text{ and } \|\nabla f(x) - \nabla f(y)\| \leq K\|x - y\| \text{ for all } x, y \in \mathbb{R}^n. \end{aligned}$$

Finally,  $f \in \mathcal{C}^{2+}$  with constant  $K$  means  $f \in \mathcal{C}^2$ , and the Hessian,  $\nabla^2 f$ , is Lipschitz continuous with Lipschitz constant  $K$ ; i.e.,

$$\begin{aligned} & f \in \mathcal{C}^{2+} \text{ with constant } K \\ \Leftrightarrow & f \in \mathcal{C}^2 \text{ and } \|\nabla^2 f(x) - \nabla^2 f(y)\| \leq K\|x - y\| \text{ for all } x, y \in \mathbb{R}^n. \end{aligned}$$

The definitions of differentiability and of directional derivatives rely on notions of limits. However, there are cases where limits are not enough to explain the behaviour of a function. For example, consider the function of a single variable  $f : \mathbb{R} \rightarrow \mathbb{R}$  defined as  $f(0) = 0$  and  $f(x) = \cos(1/x)$  otherwise. The limit of  $f(x)$  as  $x$  goes to 0 is undefined. However, one can easily find subsequences converging to 0 whose function values converge to any value

between  $-1$  and  $1$ . For instance, the sequence  $x_k = 1/(2k\pi)$  converges to  $0$  and  $f(x_k) = \cos(2k\pi) = 1$  for every integer  $k$ . The *limit inferior* and *limit superior* are used to resolve these conflicts.

Recall  $\inf S$  is the *infimum* (greatest lower bound) of the set  $S$ , and  $\sup S$  is the *supremum* (least upper bound) of the set  $S$ .

**DEFINITION 2.3 (Limit Superior and Limit Inferior).**

The *limit superior* of the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  at  $x \in \mathbb{R}^n$  is defined to be

$$\limsup_{y \rightarrow x} f(y) = \lim_{r \searrow 0} u_x(r)$$

where  $u_x(r)$  is a function that returns the least upper-bound on  $f$  over the open ball of radius  $r > 0$  centred at  $x$ :

$$u_x(r) = \sup_{y \in B_r(x)} f(y), \text{ where } B_r(x) := \{y : \|y - x\| < r\}.$$

Similarly, the *limit inferior* of  $f$  at  $x$  is defined to be

$$\liminf_{y \rightarrow x} f(y) = \lim_{r \searrow 0} \left( \inf_{y \in B_r(x)} f(y) \right), \text{ where } B_r(x) := \{y : \|y - x\| < r\}.$$

An important aspect of these limits is the fact that if  $f$  is bounded below, then  $\liminf_{y \rightarrow x} f(y)$  is a well-defined real number. Similarly if  $f$  is bounded above, then  $\limsup_{y \rightarrow x} f(y)$  is a well-defined real number. Moreover, the following inequalities are always valid

$$\liminf_{y \rightarrow x} f(y) \leq \limsup_{y \rightarrow x} f(y),$$

and equality holds if and only if  $\lim_{y \rightarrow x} f(y)$  exists. In addition, if  $\{x^k\}$  is any particular sequence that converges to  $x$ , then

$$\liminf_{y \rightarrow x} f(y) \leq \liminf_k f(x^k) \leq \limsup_k f(x^k) \leq \limsup_{y \rightarrow x} f(y).$$

Figure 2.1 illustrates some of these notions.

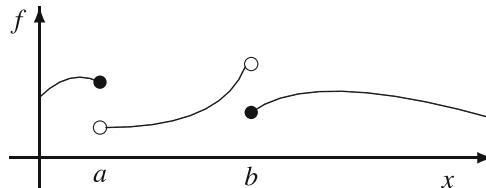


FIGURE 2.1. The function  $f$  satisfies  $\limsup_{y \rightarrow a} f(y) = f(a)$  and  $\liminf_{y \rightarrow b} f(y) = f(b)$

The next theorem is used later in the book, and its proof relies on the definitions of limit inferior and of limit superior.

**THEOREM 2.1 (Squeeze Theorem).**

Let  $f, g$ , and  $h$  be functions and  $x \in \mathbb{R}^n$ . If  $f(y) \leq g(y) \leq h(y)$  for every  $y \in B_r(x)$  for some  $r > 0$ , and if

$$\lim_{y \rightarrow x} f(y) = \lim_{y \rightarrow x} h(y) = L,$$

then  $\lim_{y \rightarrow x} g(y) = L$ .

**PROOF.** Taking limits inferior and superior yields

$$L = \lim_{y \rightarrow x} f(y) \leq \liminf_{y \rightarrow x} g(y) \leq \limsup_{y \rightarrow x} g(y) \leq \lim_{y \rightarrow x} h(y) = L,$$

and therefore all inequalities are in fact equalities.  $\square$

### 2.3. Descent Directions

Of great importance to optimization is the idea of a *descent direction*.

**DEFINITION 2.4 (Descent Direction).**

A direction  $d \in \mathbb{R}^n$  is said to be a descent direction of the function  $f$  at the point  $x \in \mathbb{R}^n$  if and only if there exists a scalar  $\bar{t} > 0$  such that

$$f(x + td) < f(x) \quad \text{for all } t \in (0, \bar{t}).$$

The role of a descent direction in optimization algorithms is fairly straightforward; if  $d \in \mathbb{R}^n$  is a descent direction of  $f$  at  $x \in \mathbb{R}^n$ , then evaluating  $f(x + td)$  for small values of  $t$  should lead to an improved estimate of the minimum value.

Descent directions are closely linked to directional derivatives in the following sense. Suppose all directional derivatives of  $f$  are well defined at  $x$ , then

- i. if  $d$  is a descent direction of  $f$  at  $x$ , then  $f'(x; d) \leq 0$ ; and
- ii. if  $f'(x; d) < 0$ , then  $d$  is a descent direction of  $f$  at  $x$ .

Notice the slight asymmetry that arises from examples such as  $f(x) = x^3$  at the point 0. Indeed,  $f'(0; d) = 0$  for every direction  $d \in \mathbb{R}$ , but any  $d < 0$  is a descent direction, and any  $d > 0$  is an ascent direction.

An important corollary is that if  $\nabla f(x)$  exists and is nonzero, then  $f$  has a descent direction at  $x$  (see Exercise 2.6), so  $x$  is not the minimiser of  $f$ . The contrapositive of this is called the *first order optimality condition*, or *Fermat's Theorem*, for unconstrained  $\mathcal{C}^1$  functions:

$$\text{if } f \in \mathcal{C}^1 \text{ and } x \in \operatorname{argmin}\{f(x) : x \in \mathbb{R}^n\}, \text{ then } \nabla f(x) = 0.$$

A point  $x \in \mathbb{R}^n$  for which  $\nabla f(x) = 0$  is said to be critical. Section 6.4 generalises this result to the situation where the function  $f$  is not differentiable.

## 2.4. Basic Properties of Sets

In Multivariate Calculus, a set is called *open* if every point is on the interior of the set, and a set is called *closed* if it contains its boundary. These informal definitions are useful for building intuition, but not sufficient for the analysis in this book. The formal definitions of open and closed sets are as follows.

**DEFINITION 2.5 (Open Set and Interior).**

A set  $S$  is open if for any  $x \in S$  there exists  $r > 0$  such that  $B_r(x) \subseteq S$ , where  $B_r(x) := \{y : \|y - x\| < r\}$ .

The interior of a set  $S$ , denoted by  $\text{int}(S)$ , is the largest open set that is contained in  $S$ .

**DEFINITION 2.6 (Closed Set and Closure).**

A set  $S$  is closed if for any convergent sequence of points  $\{x^1, x^2, \dots\}$  such that  $x^i \in S$  is such that  $\lim_{i \rightarrow \infty} x^i \in S$ , i.e., its limit belongs to the set.

The closure of a set  $S$ , denoted  $\text{cl}(S)$ , is the smallest closed set containing  $S$ .

The *open ball* of radius  $r > 0$  centred at  $x$ ,  $B_r(x)$ , has been useful in defining the limit inferior, limit superior, and open sets. Unsurprisingly, the open ball is an open set. Its counterpart, the *closed ball* of radius  $r$  centred at  $\bar{x}$ , will be denoted  $\text{cl}(B_r(\bar{x}))$  and satisfies  $\text{cl}(B_r(\bar{x})) = \{x : \|x - \bar{x}\| \leq r\}$ .

The whole space  $\mathbb{R}^n$  and the empty set  $\emptyset$  satisfy both definitions, and therefore are both open and closed. A very useful property is that a set  $S \subseteq \mathbb{R}^n$  is closed if and only if its complement  $\mathbb{R}^n \setminus S$  is open.

*Bounded* and *compact* complete the standard list of set properties.

**DEFINITION 2.7 (Bounded and Compact Sets).**

A set  $S$  is bounded if there exists a scalar  $r > 0$  such that  $\|x\| \leq r$  for all  $x \in S$ . A set  $S$  is compact if it is closed and bounded.

Compact sets are extremely powerful in optimization, for two reasons. First, given any infinite sequence  $\{x^k\}_{k=1}^\infty$  in a compact set  $S$ , there must exist an *accumulation point*. That is, there exist a point  $\hat{x} \in S$  and a *convergent subsequence*  $\{x^{k_j}\}_{j=1}^\infty$  such that  $x^{k_j} \rightarrow \hat{x}$ . The second powerful result of compact sets in optimization is the Extreme Value Theorem, which states that any continuous function attains its bound on a compact set.

**THEOREM 2.2 (Extreme Value Theorem).**

Suppose  $f \in \mathcal{C}^0$  and  $\Omega$  is a nonempty compact set. Then

$$\operatorname{argmin}_x \{f(x) : x \in \Omega\} \neq \emptyset.$$

When comparing a point  $x$  to a set  $\Omega$  it is often important to check how “far”  $x$  is from  $\Omega$ . This is formalised in the *distance function*.

**DEFINITION 2.8 (Distance Function).**

Let  $\Omega \subseteq \mathbb{R}^n$  a nonempty closed set. The distance of a point  $x$  to  $\Omega$ , denoted by  $\operatorname{dist}(x, \Omega)$ , is defined as

$$\operatorname{dist}(x, \Omega) := \min_y \{ \|x - y\| : y \in \Omega\}.$$

As long as  $\Omega$  is nonempty, the distance function is well defined.

One very special set that arises often in optimization is the *level set* of a function.

**DEFINITION 2.9 (Level Set).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ . The level set of  $f$  with level  $c$ , denoted by  $\mathcal{L}(c)$ , is

$$\mathcal{L}(c) := \{x \in \mathbb{R}^n : f(x) \leq c\}.$$

If the function  $f$  is continuous, then  $\mathcal{L}(c)$  is closed for all values of  $c$  (note that both the empty set and the space  $\mathbb{R}^n$  are closed). A very common assumption in optimization literature is that the level sets of the objective function are compact. This is useful to avoid optimizing functions such as  $f(x) = e^x$  which is bounded below but does not have a minimiser.

## 2.5. Convexity

We call a set *convex* if it contains all of the *line segments* defined using pairs of points in the set.

**DEFINITION 2.10 (Convex Set).**

A set  $\Omega$  is convex if given any two points  $x, y \in \Omega$ , and any  $\theta \in [0, 1]$  we have  $\theta x + (1 - \theta)y \in \Omega$ .

Simple rearrangements provide alternative definitions for line segments, and hence convexity:

$$\begin{aligned} \{\theta x + (1 - \theta)y : 0 \leq \theta \leq 1\} &= \{\theta_1 x + \theta_2 y : \theta_i \geq 0, \theta_1 + \theta_2 = 1\} \\ &= \{y + \theta(x - y) : 0 \leq \theta \leq 1\}. \end{aligned}$$

Basic examples of convex sets are easy to create. Indeed, both the open ball  $B_\varepsilon(x)$  and the closed ball  $\text{cl}(B_\varepsilon(x))$  are convex. Some other convex and nonconvex sets are illustrated in Figure 2.2.

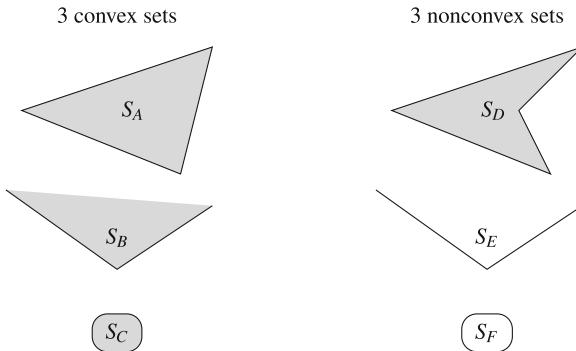


FIGURE 2.2. Convex and nonconvex sets in  $\mathbb{R}^2$  ( $S_F$  only contains the boundary)

In working with convex sets, it is useful to consider the idea of a *convex combination*.

**DEFINITION 2.11 (Convex Combination).**

A point  $z \in \mathbb{R}^n$  is a convex combination of the points in  $\{x^1, x^2, \dots, x^m\} \subseteq \mathbb{R}^n$  if there exists scalars  $\theta_1, \theta_2, \dots, \theta_m$ , such that

$$z = \sum_{i=1}^m \theta_i x^i, \text{ with } \sum_{i=1}^m \theta_i = 1, \theta_i \in \mathbb{R}, \text{ and } \theta_i \geq 0 \text{ for } i = 1, 2, \dots, m.$$

If a set is not convex, we sometimes work with the *convex hull* of the set.

**DEFINITION 2.12 (Convex Hull).**

The convex hull of a set  $\Omega$  is the smallest convex set containing  $\Omega$ , denoted by  $\text{conv}(\Omega)$ .

For any set  $\Omega$ , we have  $\Omega \subseteq \text{conv}(\Omega)$ . Moreover,  $\Omega$  is convex if and only if  $\Omega = \text{conv}(\Omega)$ . Finally, the next proposition shows that the convex hull  $\text{conv}(\Omega)$  can be constructed as the set of all convex combinations generated using points from  $\Omega$ .

**PROPOSITION 2.3.**

Let  $\Omega \subseteq \mathbb{R}^n$ . Then

$$\text{conv}(\Omega) = \left\{ y \in \mathbb{R}^n : y = \sum_{i=1}^m \theta_i x^i, x^i \in \Omega, \sum_{i=1}^m \theta_i = 1, \theta_i \geq 0, m \geq 1 \right\}.$$

This theorem is particularly useful if the set  $\Omega$  is a finite list of points.

The sets in Figure 2.2 satisfy  $S_A = \text{conv}(S_D)$  and  $S_C = \text{conv}(S_F)$  but  $S_B \neq \text{conv}(S_E)$  because the line segment joining the leftmost and rightmost points of  $S_E$  is not contained in  $S_B$ .

We next extend the definition of convexity to functions. A function is said to be convex if its *epigraph* forms a convex set.

**DEFINITION 2.13 (Convex Function).**

*The function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  is convex if and only if the set*

$$\text{epi}(f) := \{[x, \alpha]^\top \in \mathbb{R}^n \times \mathbb{R} : \alpha \geq f(x)\}$$

*is convex. The set  $\text{epi}(f)$  is called the epigraph of  $f$ .*

Equivalently, convexity for a function can be defined by examining the secant lines generated using the function.

**COROLLARY 2.4 (Characterisation of a Convex Function).**

*The function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  is convex if and only if*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) \quad \text{for all } x, y \in \mathbb{R}^n, \theta \in [0, 1].$$

Convex functions are very useful in optimization as Fermat's Theorem from Section 2.3 becomes an “if and only if” statement:

$$\begin{aligned} &\text{if } f \in \mathcal{C}^1 \text{ is convex, then } x \in \arg\min\{f(x)\} \text{ if and only if} \\ &\nabla f(x) = 0. \end{aligned}$$

While many other properties of convex sets and functions exist, this quick overview provides the general background required for the majority of this book. Other properties of convex sets and functions will be introduced as needed.

## 2.6. Simplices

Many DFO methods rely on certain collections of vectors creating a *simplex*, defined shortly. The clearest example is the Nelder-Mead algorithm of Chapter 5, where the entire method focuses on manipulation of simplices.

Less clear examples arise in Parts 3 and 4. For the class of pattern search methods from Part 3, convergence relies on the convex hull of the *poll directions* having a nonempty interior. In model-based methods, Part 4, good models rely on a *well-poised* sample set. It turns out that both of these statements can be understood in terms of a simplex. As such, it is valuable to review this geometric object before proceeding to DFO algorithms.

Recall that a convex polytope is a geometric object in  $\mathbb{R}^n$  that has flat sides. Convex polytopes can always be described as the intersection of a finite number of half-spaces, or as the solution set to a matrix inequality  $C = \{x \in \mathbb{R}^n : Ax \leq b\}$  where  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ , or as the convex hull of a finite number of points. With this background in place, we define a simplex.

**DEFINITION 2.14 (Simplex).**

A simplex in  $\mathbb{R}^n$  is a bounded convex polytope with nonempty interior and exactly  $n + 1$  vertices.

In essence, a simplex is the simplest convex polytope that has interior. For example, in  $\mathbb{R}^1$  a simplex is an interval, in  $\mathbb{R}^2$  a simplex is a triangle, and in  $\mathbb{R}^3$  a simplex is a triangular-based pyramid. In Figure 2.2,  $S_A$  is a simplex in  $\mathbb{R}^2$ , but  $S_B$  is not a simplex.

It might be easy to assume that a simplex is the convex hull of  $n + 1$  points in  $\mathbb{R}^n$ . However, note that this is not quite correct, as the convex hull of

$$\{[1, 0, 0]^\top, [-1, 0, 0]^\top, [0, 1, 0]^\top, [0, -1, 0]^\top\}$$

does not form a simplex in  $\mathbb{R}^3$ , as it lacks an interior. Nonetheless, the convex hull of  $n + 1$  points in  $\mathbb{R}^n$  will often create a simplex. Moreover, there is a very simple test to check if the convex hull will form a simplex.

**THEOREM 2.5 (Simplex Tests).**

Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  be a set of  $n + 1$  points in  $\mathbb{R}^n$ . Then the following are equivalent:

- i.  $\text{conv}(\mathbb{Y})$  is a simplex.
- ii. The set  $\{(y^1 - y^0), (y^2 - y^0), \dots, (y^n - y^0)\}$  is linearly independent.
- iii. The matrix  $L = [(y^1 - y^0) \quad (y^2 - y^0) \quad \dots \quad (y^n - y^0)]$  is invertible.
- iv. The matrix  $L = [(y^1 - y^0) \quad (y^2 - y^0) \quad \dots \quad (y^n - y^0)]$  satisfies  $\det(L) \neq 0$ .

**PROOF.** Recall,  $\text{conv}(\mathbb{Y})$  is a simplex if and only if  $\text{conv}(\mathbb{Y})$  has  $n + 1$  vertices and  $\text{int}(\text{conv}(\mathbb{Y})) \neq \emptyset$ . Since shifting a polytope will not alter the number of vertices and since  $\text{int}(\text{conv}(\mathbb{Y} - y^0)) = \text{int}(\text{conv}(\mathbb{Y})) - y^0$ , we see that

$$\begin{aligned} & \text{conv}(\mathbb{Y}) \text{ is a simplex} \\ \Leftrightarrow & \text{conv}(\mathbb{Y} - y^0) = \text{conv}(\{0, y^1 - y^0, y^2 - y^0, \dots, y^n - y^0\}) \text{ is a simplex.} \end{aligned}$$

We next show that i.  $\Leftrightarrow$  ii.

( $\Rightarrow$ ) Suppose  $\text{int}(\text{conv}(\mathbb{Y} - y^0)) \neq \emptyset$ , then there exists  $x \in \text{conv}(\mathbb{Y} - y^0)$  and  $r > 0$  such that  $B_r(x) \subseteq \text{conv}(\mathbb{Y} - y^0)$ . Since

$$\text{conv}(\mathbb{Y} - y^0) = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^n \theta_i(y^i - y^0), \theta_i \geq 0, \sum_{i=1}^n \theta_i = 1 \right\},$$

it is easy to see that  $\text{span}(\text{conv}(\mathbb{Y} - y^0)) = \text{span}(\mathbb{Y} - y^0)$ . Therefore we have the following relations

$$\mathbb{R}^n = \text{span}(B_r(x)) \subseteq \text{span}(\text{conv}(\mathbb{Y} - y^0)) = \text{span}(\mathbb{Y} - y^0) \subseteq \mathbb{R}^n.$$

Thus,

$$\text{span}(\{0, (y^1 - y^0), (y^2 - y^0), \dots, (y^n - y^0)\}) = \mathbb{R}^n,$$

which implies  $\{(y^1 - y^0), (y^2 - y^0), \dots, (y^n - y^0)\}$  is a basis of  $\mathbb{R}^n$  and therefore linearly independent.

( $\Leftarrow$ ) If  $\{(y^1 - y^0), (y^2 - y^0), \dots, (y^n - y^0)\}$  is linearly independent, then it is a linearly independent set of  $n$  vectors, and therefore a basis of  $\mathbb{R}^n$ . Exercise 2.13 therefore provides  $\text{conv}(\{0, (y^1 - y^0), (y^2 - y^0), \dots, (y^n - y^0)\}) = \text{conv}(\mathbb{Y} - y^0)$  has interior. This implies  $\text{conv}(\mathbb{Y} - y^0)$  has at least  $n + 1$  vertices. As  $\text{conv}(\mathbb{Y} - y^0)$  is the convex hull of  $n + 1$  points, it has at most  $n + 1$  vertices, so  $\text{conv}(\mathbb{Y} - y^0)$  is a simplex.

The remaining equivalences are standard linear algebra results.  $\square$

Consider the set  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$ . If  $\text{conv}(\mathbb{Y})$  is a simplex, then we say  $\mathbb{Y}$  *forms a simplex*. If  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  and  $\text{conv}(\mathbb{Y})$  is not a simplex, then we sometimes say  $\mathbb{Y}$  forms a *degenerate simplex*. In DFO algorithms, degenerate sets cause challenges because they do not effectively span the entire space. Numerically, even if a set is not degenerate, DFO algorithms will have difficulty if the sets are “almost degenerate”. There are many different methods to measure how close to being degenerate a simplex is, one of the most straightforward is the *normalised volume*. We begin by defining the *volume* of a simplex.

**DEFINITION 2.15 (Volume of a Simplex).**

Suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  forms a simplex in  $\mathbb{R}^n$ . Define the matrix  $L = [(y^1 - y^0) \quad (y^2 - y^0) \quad \dots \quad (y^n - y^0)]$ . Then the volume of  $\text{conv}(\mathbb{Y})$  is

$$\text{vol}(\text{conv}(\mathbb{Y})) := \frac{|\det(L)|}{n!}.$$

For ease of writing, it is common to use  $\text{vol}(\mathbb{Y}) = \text{vol}(\text{conv}(\mathbb{Y}))$ .

A very useful property of this definition is that the volume of a simplex is independent of the order of its elements. This is a consequence of the linear algebra facts on the determinant of a matrix  $M \in \mathbb{R}^{n \times n}$ :

- i. Replacing a column of  $M$  by a constant multiple of that column, multiplies the determinant by the same constant.
- ii. Adding a multiple of a column of  $M$  to a different column of  $M$  does not change the determinant.
- iii. Permuting two columns of  $M$  changes the sign of the determinant.

In our context, adding  $-(y_j - y_0)$  to every column of  $L$  but the  $j$ -th produces the matrix

$$L' = [(y^1 - y^j) \quad \dots \quad (y^{j-1} - y^j) \quad (y^j - y_0) \quad (y^{j+1} - y^j) \quad \dots \quad (y^n - y^j)]$$

with  $\det(L) = \det(L')$ . Now, reorder the elements of  $\mathbb{Y}$  so that the first one is  $y^j$  instead of  $y^0$ . Then, the determinant used to compute the volume would involve the columns of  $L'$  rather than those of  $L$ . But the absolute value of the determinant remains the same.

Of course, the volume of  $\text{conv}(\mathbb{Y})$  does not give a good measure of how close a simplex is to degenerate, as it is very easy to create a long flat simplex with very large volume. We need to relate the *diameter* and volume of the simplex to create a normalised volume.

**DEFINITION 2.16 (Diameter of a Simplex).**

Suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  forms a simplex in  $\mathbb{R}^n$ . Then the diameter of  $\text{conv}(\mathbb{Y})$  is

$$\text{diam}(\text{conv}(\mathbb{Y})) := \max\{\|y^i - y^j\| : y^i \in \mathbb{Y}, y^j \in \mathbb{Y}\}.$$

For ease of writing, it is common to use  $\text{diam}(\mathbb{Y}) = \text{diam}(\text{conv}(\mathbb{Y}))$ .

**DEFINITION 2.17 (Normalised Volume of a Simplex).**

Suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  forms a simplex in  $\mathbb{R}^n$ . Then the normalised volume of  $\text{conv}(\mathbb{Y})$  is

$$\text{von}(\text{conv}(\mathbb{Y})) := \frac{\text{vol}(\mathbb{Y})}{(\text{diam}(\mathbb{Y}))^n}.$$

For ease of writing, it is common to use  $\text{von}(\mathbb{Y}) = \text{von}(\text{conv}(\mathbb{Y}))$ .

An easy calculation shows

$$\text{von}(\text{conv}(\mathbb{Y})) = \text{vol}\left(\frac{\mathbb{Y}}{\text{diam}(\mathbb{Y})}\right),$$

which justifies the power of  $n$  in the denominator of the normalised volume formula.

**EXAMPLE 2.1.** Let us consider the volume and normalised volume of some simplices in  $\mathbb{R}^2$ .

Consider  $\mathbb{Y}^1 = \{[0, 0]^\top, [1, 0]^\top, [0, 1]^\top\}$ . Then the volume, diameter, and normalised volume of  $\mathbb{Y}^1$  are

$$\text{vol}(\mathbb{Y}^1) = \frac{1}{2}, \quad \text{diam}(\mathbb{Y}^1) = \sqrt{2}, \quad \text{von}(\mathbb{Y}^1) = \frac{1}{4}.$$

Consider  $\mathbb{Y}^2 = \{[0, 0]^\top, [2, 0]^\top, [0, 2]^\top\}$ . Notice that  $\mathbb{Y}^2$  is essentially  $\mathbb{Y}^1$  with all of its side lengths doubled. The volume, diameter, and normalised volume of  $\mathbb{Y}^2$  are

$$\text{vol}(\mathbb{Y}^2) = 2, \quad \text{diam}(\mathbb{Y}^2) = 2\sqrt{2}, \quad \text{von}(\mathbb{Y}^2) = \frac{1}{4}.$$

Although the volume and diameter changed, the normalised volume is the same.

Consider  $\mathbb{Y}^3 = \{[0, 0]^\top, [\frac{1}{10}, 0]^\top, [0, 10]^\top\}$ . Here, we have stretched  $\mathbb{Y}^1$  along the  $y$ -axis, but compensated by shrinking on the  $x$ -axis. The volume, diameter, and normalised volume of  $\mathbb{Y}^3$  are

$$\text{vol}(\mathbb{Y}^3) = \frac{1}{2}, \quad \text{diam}(\mathbb{Y}^3) = \sqrt{100.01}, \quad \text{von}(\mathbb{Y}^3) = \frac{1}{200.02}.$$

While the volume is unchanged, the normalised volume has decreased significantly. (We will see in Chapter 9 that this is a very poor-shaped simplex for building models in a DFO context.)

Finally, consider  $\mathbb{Y}^4 = \{[0, 0]^\top, [1, 0]^\top, [1/2, 1]^\top\}$ . Here, we have reshaped  $\mathbb{Y}^1$  to become more symmetric. The volume, diameter, and normalised volume of  $\mathbb{Y}^4$  are

$$\text{vol}(\mathbb{Y}^4) = \frac{1}{2}, \quad \text{diam}(\mathbb{Y}^4) = \sqrt{1.25}, \quad \text{von}(\mathbb{Y}^4) = \frac{2}{5}.$$

The normalised volume has increased, while the volume remains the same as  $\mathbb{Y}^1$ . (This is a well-shaped simplex for DFO.) All of these simplices are illustrated in Figure 2.3.

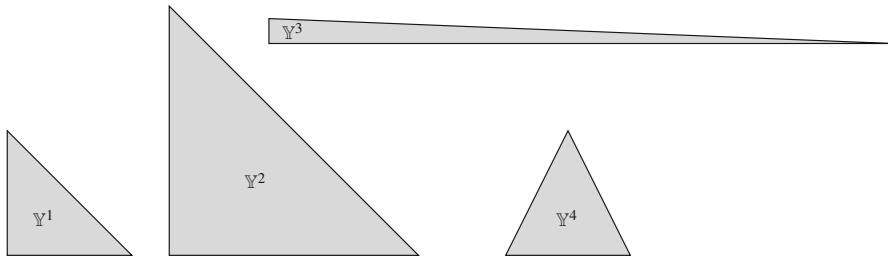


FIGURE 2.3. The simplices formed from  $\mathbb{Y}^1, \mathbb{Y}^2, \mathbb{Y}^3$ , and  $\mathbb{Y}^4$  in Example 2.1

One final, very useful, metric that relates to simplices is the *approximate diameter*. We present a definition that is not restricted to simplices, but to any finite set of points in  $\mathbb{R}^n$ . This generality will be exploited in Chapter 9.

**DEFINITION 2.18 (Approximate Diameter of a Set).**

Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\}$  be a finite ordered set of points  $\mathbb{R}^n$ . The approximate diameter of  $\text{conv}(\mathbb{Y})$  is

$$\overline{\text{diam}}(\text{conv}(\mathbb{Y})) := \max\{\|y^i - y^0\| : y^i \in \mathbb{Y}\}.$$

For ease of writing, it is common to use  $\overline{\text{diam}}(\mathbb{Y}) = \overline{\text{diam}}(\text{conv}(\mathbb{Y}))$ .

The approximate diameter of a simplex is much quicker to calculate than the true diameter, but its value depends on the order of the vertices. Using

the triangle inequality, the approximate diameter and the true diameter of a simplex  $\mathbb{Y}$  relate through a simple formula,

$$\overline{\text{diam}}(\mathbb{Y}) \leq \text{diam}(\mathbb{Y}) \leq 2 \overline{\text{diam}}(\mathbb{Y}).$$

These bounds make the approximate diameter ideal for many DFO purposes.

## EXERCISES

---

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 2.1.** Let

$$A = \begin{bmatrix} 2 & 2 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 5 \end{bmatrix}.$$

Without using a computer do the following.

- a) Find  $\det(A)$ .
- b) Find  $\|A\|_2$  and  $\|A\|_F$ .
- c) Find  $A^{-1}$ .

**EXERCISE 2.2.** Let  $f(x) = e^{x_1+2x_2+3x_3+4x_4+5x_5}$ .

- a) Find  $\nabla f(0, 0, 0, 0, 0)$  and  $\nabla f(5, 4, 3, 2, 1)$ .
- b) Let  $x = [0, 0, 0, 0, 0]^\top$  and  $v = [5, 4, 3, 2, 1]^\top$ . Find  $f'(x; v)$ .
- c) Let  $x = [5, 4, 3, 2, 1]^\top$  and  $v = [0, 0, 0, 0, 0]^\top$ . Find  $f'(x; v)$ .

**EXERCISE 2.3.** + State whether each of the following functions is Lipschitz continuous or not, and if it is convex or not. If it is Lipschitz continuous, then provide a bound for the Lipschitz constant (with proof).

- a)  $f : \mathbb{R}^n \mapsto \mathbb{R}$ ,  $f(x) = \|x\|$ .
- b)  $f : \mathbb{R}^n \mapsto \mathbb{R}$ ,  $f(x) = \|x\|^2$ .
- c)  $f : \mathbb{R}^n \mapsto \mathbb{R}$ ,  $f(x) = \alpha^\top x$ , where  $\alpha$  is a fixed vector in  $\mathbb{R}^n$ .

**EXERCISE 2.4.** Let

$$f(x) = \begin{cases} \sin(1/x) & \text{if } x \neq 0, \\ c & \text{if } x = 0. \end{cases}$$

- a) Compute  $\liminf_{y \rightarrow 0} f(y)$ .
- b) Compute  $\limsup_{y \rightarrow 0} f(y)$ .
- c) Is it possible to make  $f$  continuous? (Provide how, or show why not.)

**EXERCISE 2.5.** + Determine the interior of the set of descent directions at  $x = [1, 1, 1]^\top$  of the function  $f(x) = e^{x_1} + 5x_2 - (x_1 x_3)^2$ .

**EXERCISE 2.6.** Suppose  $f \in \mathcal{C}^1$  and  $\nabla f(x) \neq 0$  at some  $x \in \mathbb{R}^n$ .

- a) Prove that  $f$  has a descent direction  $d \in \mathbb{R}^n$  at  $x$ .
- b) Prove that if  $t > 0$  is a sufficiently small scalar, then  $f(x + td) < f(x)$ , where  $d$  is the direction from question a).

**EXERCISE 2.7.**  $\square$  Consider  $f(x, y) = \max\{f_1(x, y), f_2(x, y)\}$  where  $f_1(x, y) = x + y^2$  and  $f_2(x, y) = x^2 + y$ .

- Plot the contour diagram of  $f$ ; i.e., graph the level sets  $\mathcal{L}(c)$  for a variety of values of  $c$ .
- Determine all descent directions at the point  $[2, 0]^\top$ .
- Determine all descent directions at the point  $[1, 1]^\top$ .
- How do your solutions relate to  $\nabla f_1$  and  $\nabla f_2$ ?

**EXERCISE 2.8.** For each of the following sets state which of the following properties hold: open, closed, bounded, and/or compact. Note that sets may have more than one property, or none of the properties.

- $\{x \in \mathbb{R} : 0 \leq x \leq 1\}$ .
- $\mathbb{R}^n$ .
- The empty set  $\emptyset$ .
- $\{\lambda \in \mathbb{R}^n : \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0\}$ .
- $\{x \in \mathbb{R}^3 : (x_1)^2 + (x_2)^2 \leq (x_3)^2, x_3 > 0\}$ .

**EXERCISE 2.9.** Describe the closure and interior of the following sets.

- $\mathbb{R}^n$ .
- The empty set  $\emptyset$ .
- $C = \{x : 1 \leq \sum_{i=1}^n (x_i)^2 < 9\}$ .
- $D = \{\lambda \in \mathbb{R}^n : \sum_{i=1}^n \lambda_i = 1, \lambda_i \geq 0\}$ .
- $E = \{x \in \mathbb{R}^3 : (x_1)^2 + (x_2)^2 \leq (x_3)^2, x_3 > 0\}$ .

**EXERCISE 2.10.** + Let  $f(x) = e^{\cos(\|x\|) + \|x\|}$ ,  $x \in \mathbb{R}^n$ .

- Determine the level set  $\mathcal{L}(c)$ .
- Prove  $\mathcal{L}(e)$  is compact.
- Explain why  $\operatorname{argmin}_x \{f(x) : x \in \mathbb{R}^n\}$  must be nonempty.

**EXERCISE 2.11.** Prove that the following sets are convex.

- The open ball of radius  $r$  centred at  $\bar{x}$ :  $B_r(\bar{x}) = \{x : \|x - \bar{x}\| < r\}$ .
- The closed ball of radius  $r$  centred at  $\bar{x}$ :  $\operatorname{cl}(B_r(\bar{x})) = \{x : \|x - \bar{x}\| \leq r\}$ .
- A hyper-rectangle:  $\{x \in \mathbb{R}^n : \ell_i \leq x_i \leq u_i\}$  where  $\ell_i < u_i$  for  $i = 1, 2, \dots, n$ .

**EXERCISE 2.12.** + This question studies convexity connections between a function and its level sets.

- Prove that all level sets of a convex function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  are convex sets.
- Give an example of a non-convex function whose level sets are convex.

**EXERCISE 2.13.** + Let  $C \subseteq \mathbb{R}^n$  be a convex set. Prove that  $\operatorname{int}(C) \neq \emptyset$  if and only if there exists a point  $x \in C$  and a basis  $\{b^1, b^2, \dots, b^n\}$  such that  $x + b^i \in C$  for all  $i = 1, 2, \dots, n$ .

**EXERCISE 2.14.** Compute the volume and the normalised volume of each of the following. Determine if the set is a simplex, a degenerate simplex or an almost degenerated simplex (for the purpose of this question, the threshold for qualifying as being almost degenerate is a normalised volume of 1/100).

- a)  $\text{conv}(\{[0, 0, 0]^\top, [0, 0, 1]^\top, [0, 1, 0]^\top, [1, 0, 0]^\top\})$ .
- b)  $\text{conv}(\{[4, 0, 1]^\top, [0, 0, 1]^\top, [0, 0, 5]^\top, [0, 4, 1]^\top\})$ .
- c)  $\text{conv}(\{[2, 2, 2]^\top, [3, 1, 4]^\top, [2, 4, -6]^\top, [4, -1, 6]^\top\})$ .

**EXERCISE 2.15.** Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  form a simplex. Prove  $\overline{\text{diam}}(\mathbb{Y}) \leq \text{diam}(\mathbb{Y}) \leq 2\overline{\text{diam}}(\mathbb{Y})$ .

---



**Chapter 2 Project: A Uniform Simplex.** Find all sets of points  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  in  $\mathbb{R}^n$  that simultaneously satisfy the following properties:

- a)  $\mathbb{Y}$  forms a simplex in  $\mathbb{R}^n$ .
- b)  $\|y^i\|_2 = 1$  for every  $i = 0, 1, \dots, n$ .
- c)  $\sum_{i=0}^n y^i = 0$ .
- d) Let  $\pi^i$  be the line segment joining the origin and  $y^i$  for  $i = 0, 1, \dots, n$ .  
The angle between  $\pi^i$  and  $\pi^j$  is independent of  $i$  and  $j$ .
- e) The matrix  $M = [y^1 \ y^2 \ \dots \ y^n]$  is upper triangular, i.e.,  $M_{i,j} = 0$  for all  $n \geq i > j \geq 1$
- f) Each element on the main diagonal of  $M$  is strictly positive, i.e.,  $M_{i,i} > 0$  for all  $i = 1, 2, \dots, n$ .

# *The Beginnings of DFO Algorithms*

In this chapter, we explore some naive approaches to solving BBO problems, and explain why they are not acceptable. This will lead to a better understanding of why DFO is preferred for real applications and provide some foundational material that is used throughout this book.

## 3.1. Exhaustive Search

The first algorithmic approach that we describe is possibly the first that comes to mind. Its presentation requires the following definitions.

### DEFINITION 3.1 (Dense Subset).

*A set  $S$  is said to be a dense subset of  $\Omega$  if  $S \subseteq \Omega$ , and given any point  $x \in \Omega$  and any distance threshold  $\delta > 0$ , there exists a point  $\tilde{x} \in S$  such that  $\|x - \tilde{x}\| < \delta$ .*

Consequently,  $S$  is a dense subset of a closed set  $\Omega$  if the closure of  $S$  equals  $\Omega$ . A set is said to be *countable* if it can be ordered: i.e.,  $S = \{x^0, x^1, \dots\}$ . The classic example of a countable set is the set of integers, where we can create the ordering  $\mathbb{Z} = \{0, +1, -1, +2, -2, +3, -3, \dots\}$ . The rational numbers are also countable, although creating the ordering requires a little more creativity.

Given these definitions, we can present the most naive optimization approach of all, the exhaustive search for  $\min_x \{f(x) : x \in \Omega\}$ .

---

**ALGORITHM 3.1.** Exhaustive search (ES)

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and  $\Omega \subseteq \mathbb{R}^n$

0. Initialisation

|  |                                      |
|--|--------------------------------------|
| $S = \{x^0, x^1, x^2, \dots\}$         | a countable dense subset of $\Omega$ |
| $k \leftarrow 0$                       | iteration counter                    |
| $x_{\text{best}} = x^0 \in S$          | best point found so far              |
| $f_{\text{best}} = f(x_{\text{best}})$ | best objective function value so far |

1. Update

|   |
|---|
| evaluate $f(x^{k+1})$   |
| if $f(x^{k+1}) < f_{\text{best}}^k$ , then  |
| update $x_{\text{best}} \leftarrow x^{k+1}$ , $f_{\text{best}} \leftarrow f(x_{\text{best}})$ |
| increment $k \leftarrow k + 1$ and go to 1  |

---

The convergence of ES is fairly straightforward, but gives us a chance to demonstrate how some of our basic assumptions will be used. In particular, we will assume  $f$  is continuous,  $f \in \mathcal{C}^0$ , and the problem has at least one solution,  $\operatorname{argmin}_x \{f(x) : x \in \Omega\} \neq \emptyset$ . We also postpone to the next subsection the notion of an algorithmic stopping criteria. For the convergence analysis, we make the unrealistic hypothesis that the ES algorithm is run indefinitely.

**THEOREM 3.1 (Convergence of ES).**

Suppose  $f \in \mathcal{C}^0$ ,  $\Omega \subseteq \mathbb{R}^n$  and  $X^* = \operatorname{argmin}_x \{f(x) : x \in \Omega\} \neq \emptyset$ .

Denote the value  $f_{\text{best}}$  at iteration  $k$  of ES by  $f_{\text{best}}^k$  and the point  $x_{\text{best}}$  at iteration  $k$  by  $x_{\text{best}}^k$ . Then

$$\lim_{k \rightarrow \infty} f_{\text{best}}^k = f^* := \min_x \{f(x) : x \in \Omega\}$$

and moreover, if  $\Omega$  is compact, then

$$\liminf_{k \rightarrow \infty} \operatorname{dist}(x_{\text{best}}^k, X^*) = 0.$$

**PROOF.** By definition of optimality,  $f_{\text{best}}^k \geq f^*$  for all  $k$ . Next, select any optimal solution  $x^* \in X^*$ . Since  $f \in \mathcal{C}^0$ , given any  $\varepsilon > 0$ , there exists  $\delta > 0$  such that

$$x \in B_\delta(x^*) \Rightarrow f(x^*) - \varepsilon < f(x) < f(x^*) + \varepsilon.$$

Since  $S$  is dense, there exists  $x^k \in S$  such that  $x^k \in B_\delta(x^*)$ . So, as soon as iteration  $k$  is complete, we must have

$$f^* \leq f_{\text{best}}^k \leq f(x^k) < f(x^*) + \varepsilon.$$

As  $\varepsilon$  was arbitrary, we must have  $\lim_{k \rightarrow \infty} f_{\text{best}}^k = f^*$ .

Suppose now that  $\Omega$  is compact. As such, all points  $x_{\text{best}}^k$  lie in a compact set, so there exists a convergent subsequence  $x_{\text{best}}^{k,i} \rightarrow \bar{x}$ . Since  $f \in \mathcal{C}^0$ , we have  $\lim_{k \rightarrow \infty} f(x_{\text{best}}^{k,i}) = f(\bar{x})$ . But we already know  $\lim_{k \rightarrow \infty} f_{\text{best}}^k = f^*$ . Hence,  $f(\bar{x}) = f^*$ , so  $\bar{x} \in X^*$ . Thus  $\liminf_{k \rightarrow \infty} \text{dist}(x_{\text{best}}^k, X^*) = 0$ .  $\square$

The proof of Theorem 3.1 used the continuity of  $f$  twice. If  $f$  is not continuous, then it is fairly easy to create an example where ES fails. For example, set

$$f(x) = \begin{cases} 1 & \text{if } x \neq \pi \\ 0 & \text{if } x = \pi, \end{cases}$$

then use any countable dense sequence that does not include  $\pi$ .

The proof of Theorem 3.1 also used  $\underset{x}{\operatorname{argmin}}\{f(x) : x \in \Omega\} \neq \emptyset$ . To see the importance of this assumption, imagine what would happen if ES (or any algorithm) was applied to minimise  $f(x) = x$  over  $x \in \mathbb{R}$ .

Finally, the last part of the proof of Theorem 3.1 used the hypothesis that  $\Omega$  is compact. If  $\Omega$  is not compact, then it is possible to create an example where  $x_{\text{best}}^k$  does not converge to a minimiser, despite  $f_{\text{best}}^k$  converging to  $f^*$  (see Exercises 3.1 and 3.2).

## 3.2. Grid Search

While ES is proven to converge, it should be fairly clear that it requires infinitely many function evaluations. Moreover, if ES is terminated after a finite number of iterations, then it is possible that  $x_{\text{best}}$  is nowhere near a true minimiser. Finally, producing a countable dense subset of  $\Omega$  is a nontrivial task.

If the constraint set  $\Omega$  is bounded, then a tempting way to approach the problem  $\underset{x}{\operatorname{min}}\{f(x) : x \in \Omega\}$  is to simply evaluate the objective function value  $f$  at a large number of points located on a grid on the space of variables. For example, if  $\Omega = [\ell, u] = \{x \in \mathbb{R}^n : \ell_i \leq x_i \leq u_i, i = 1, 2, \dots, n\}$  is a box, then one could discretise each variable  $x_i$  so that it takes  $p$  equidistant values in the interval  $[\ell_i, u_i]$ . Then, one would evaluate  $f$  at all the  $p^n$  grid points  $\mathcal{G}$ , and return the one with the lowest objective function value. This is summarised in Algorithm 3.2.

**ALGORITHM 3.2.** Grid search (GS) 

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and  $[\ell, u] \subset \mathbb{R}^n$

0. Initialisation

|  $p \in \mathbb{N}$  with  $p \geq 2$  number of grid points for each variable

1. Sample  $f$  at all grid points

| define  $\delta_i = (u_i - \ell_i)/(p - 1)$  for each  $i \in \{1, 2, \dots, n\}$   
 create  $G = \{x \in [\ell, u] : x_i = \ell_i + z\delta_i, i = 1, 2, \dots, n, z = 0, 1, \dots, p - 1\}$   
 choose  $x_{\text{best}} \in \operatorname{argmin}\{f(t) : t \in G\}$

---

Notice that if GS is repeated for ever increasing values of  $p$ , then we generate a countable dense subset of  $[\ell, u]$ , so the convergence proof of ES applies. However, we can go one step further with GS. If  $f$  is Lipschitz continuous, then the final accuracy of GS can be quantified in terms of the Lipschitz constant and the number of grid points used per variable.

**THEOREM 3.2** (Convergence of GS).

Suppose  $f \in \mathcal{C}^{0+}$  with constant  $K$ . Suppose GS is used to minimise  $f$  over the box constraint  $x \in [\ell, u] \subseteq \mathbb{R}^n$  using  $p \in \mathbb{N}$  grid points for each variable. The optimal value  $f^* = \min_x \{f(x) : x \in [\ell, u]\}$  satisfies

$$f_{\text{best}} - K\sqrt{n}\frac{\delta}{2} \leq f^* \leq f_{\text{best}},$$

where  $f_{\text{best}} = f(x_{\text{best}})$ .

**PROOF.** Exercise 3.3.  $\square$

When applying Theorem 3.2, it is important to remember that  $p$  is the number of grid points per variable. The total number of function evaluations in dimension  $n$  will be  $p^n$ . This value grows exponentially with dimension, making GS extremely inefficient, even for low dimension problems. For illustration, let us apply the GS algorithm to the smooth and nonsmooth variants of the rheology parameter fit optimization problem from Section 1.3.3.

**EXAMPLE 3.1.** The first difficulty in applying the GS algorithm to the rheology parameter fit problem is that the three variables are unbounded. We address this by introducing artificial bounds on the variables.

We begin with the “baseline solution”  $[\eta_0, \lambda, \beta]^\top = [5200, 140, 0.38]^\top$ , whose function values are

$$\hat{g}(5200, 140, 0.38) = 462.446 \quad \text{and} \quad \check{g}(5200, 140, 0.38) = 21,000.5.$$

Since the magnitude of the three parameters are very different, we rescale the problem by defining new functions  $\hat{f}$  and  $\check{f}$ , via

$$\hat{f}(x) := \hat{g}(520x_1, 14x_2, 0.038x_3) \quad \text{and} \quad \check{f}(x) := \check{g}(520x_1, 14x_2, 0.038x_3).$$

The baseline solution for the rescaled problem is simply  $x^0 := [10, 10, 10]^\top$ .

We now introduce bounds on the variables, specifically we set  $\ell = [0, 0, 0]^\top$  and  $u = [20, 20, 20]^\top$  and consider the pair of optimization problems

$$(3.1) \quad \min_{x \in \mathbb{R}^3} \left\{ \hat{f}(x) : x \in [\ell, u] \right\},$$

$$(3.2) \quad \min_{x \in \mathbb{R}^3} \left\{ \check{f}(x) : x \in [\ell, u] \right\}.$$

Table 3.1 shows the result of running the GS algorithm using different values of the parameter  $p$ . The values  $p = 10, 22$ , and  $47$  are chosen so that GS requires approximately  $10^3, 10^4$ , and  $10^5$  grid points. The value  $p = 2001$  is chosen so that GS seeks two decimals of accuracy in each coordinate. Using  $p = 2001$  results in more than  $8 \times 10^9$  grid points, which is an unacceptably large number in a BFO/BBO context.

TABLE 3.1. The approximate number of function evaluations required and results of GS applied to the rheology optimization problem for five different input values of  $p$

| $p$  | $n_{\text{eval}}$<br>(approx) | Nonsmooth problem                    |                            | Smooth problem                         |                              |
|------|-------------------------------|--------------------------------------|----------------------------|--|------------------------------|
|      |                               | $x_{\text{best}}^\top$ for $\hat{f}$ | $\hat{f}(x_{\text{best}})$ | $x_{\text{best}}^\top$ for $\check{f}$ | $\check{f}(x_{\text{best}})$ |
| 8    | 512                           | [11.43, 11.43, 8.57]                 | 278.219                    | [11.43, 11.43, 8.57]                   | 10454.332                    |
| 10   | $10^3$                        | [11.11, 11.11, 8.89]                 | 143.267                    | [11.11, 11.11, 8.89]                   | 4538.410                     |
| 22   | $10^4$                        | [10.48, 9.52, 8.57]                  | 176.952                    | [11.43, 12.38, 9.52]                   | 3992.124                     |
| 47   | $10^5$                        | [9.13, 7.83, 8.70]                   | 65.288                     | [9.13, 7.83, 8.70]                     | 713.197                      |
| 2001 | $8 \times 10^9$               | [9.49, 8.38, 8.71]                   | 33.225                     | [9.48, 8.36, 8.71]                     | 172.220                      |

We conclude the example with two observations. First, notice that the solution to  $\hat{f}$  with  $10^4$  grid points is worse than that with  $10^3$  points. Second, while  $\hat{f}$  and  $\check{f}$  yield similar points for  $x_{\text{best}}^\top$ , they are not identical.

### 3.3. Coordinate Search

We have seen that the GS algorithm requires an enormous number of evaluations to get a solution whose objective function value is close to the optimal one. In most BBO applications, function evaluations are considered very time consuming. For example, at 1 millisecond per function evaluation, the  $8 \times 10^9$  function evaluations required to find 2 digits of accuracy in Example 3.1 would take over 90 days.

A simple, but considerably more effective method to improve on GS is the Coordinate Search algorithm. While CS is not the most efficient method for solving BBO problems, it shares some similar features with more evolved algorithm, and motivates the design of more sophisticated methods.

At each iteration (denoted by the integer  $k$ ) the CS algorithm selects the best known candidate solution known as the incumbent solution, and then examines points along the coordinate directions  $\pm e_i \in \mathbb{R}^n$  at a fixed step-length  $\delta^k > 0$ . If improvement is found, then the incumbent solution is updated; otherwise, the step-length is decreased.

Unlike GS where one needs to supply bounds for each variable,  $x \in [\ell, u]$ , the CS algorithm requires only one initial point  $x^0 \in \mathbb{R}^n$ . If a constraint set  $\Omega$  is provided, then CS can be applied by setting the objective function value to  $\infty$  whenever the constraint is violated.

A formalisation of CS is present in Algorithm 3.3.

---

**ALGORITHM 3.3. Coordinate search (CS)**

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and starting point  $x^0 \in \mathbb{R}^n$

0. Initialisation

|  |                     |
|--|---------------------|
| $\delta^0 \in (0, \infty)$               | initial step length |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance  |
| $k \leftarrow 0$                         | iteration counter   |

1. Poll

|  |  |
|--|--|
| if $f(t) < f(x^k)$ for some $t \in P^k := \{x^k \pm \delta^k e_i : i = 1, 2, \dots, n\}$ | set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \delta^k$              |
| otherwise  | set $x^{k+1} \leftarrow x^k$ and $\delta^{k+1} \leftarrow \frac{1}{2}\delta^k$ |

2. Termination

|   |  |
|---|--|
| if $\delta^{k+1} \geq \epsilon_{\text{stop}}$ | increment $k \leftarrow k + 1$ and go to 1 |
| otherwise                                     | stop                                       |

---

Iteration  $k$  and iterate  $x^k$  of CS are labelled either as successful or unsuccessful. Successful iterations are those where a new incumbent solution is generated. Unsuccessful ones are those where  $x^k$  is shown to be an optimizer of  $f$  with respect to the discrete set  $P^k$ , called the *poll set*.

**EXAMPLE 3.2.** Figure 3.1 illustrates three iterations of a CS algorithm on a problem with  $n = 2$  variables. The initial point is  $x^0 = [2, 2]^\top$  and is represented by the dark circle, The initial step length parameter  $\delta^0$  is chosen to be equal to one.

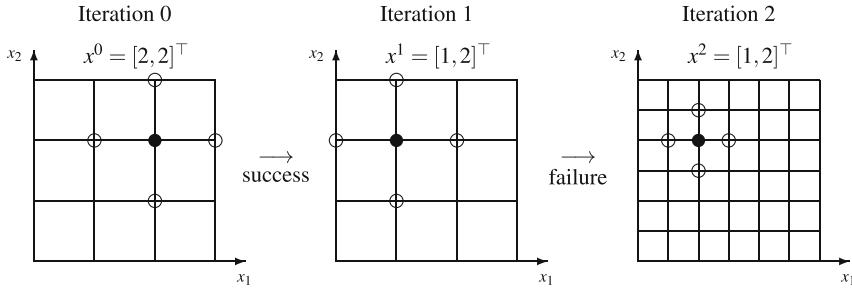


FIGURE 3.1. A successful and an unsuccessful iteration of a CS algorithm

At the first iteration, the poll set  $P^0$  is represented by the four open circles. The trial point  $[1, 2]^\top$  has an objective function value that is less than  $f(x^0)$ , leading to a successful iteration. Iteration 1 starts with  $x^1 = [1, 2]^\top$ . However, iteration 1 fails in identifying a trial point in the poll set  $P^1$  with a lower objective function value. Iteration 2 starts with  $x^2 = x^1$ , but the step size parameter is cut in half, so the new poll set  $P^2$  uses points that are closer to the incumbent solution.

---

In  $\mathbb{R}^2$ , the CS algorithm is sometimes referred to as *Compass Search*, since the directions used can be seen as the four cardinal directions.

An important property of the CS algorithm is that the number of trial points at each iteration grows linearly with the number of variables. For an  $n$ -dimensional problem, each iteration of the algorithm explores exactly  $2n$  directions. Unfortunately, the number of iterations required to approach a local solution typically increases with  $n$ .

In theory, the CS algorithm can be run indefinitely. In fact, the convergence analysis below studies the behaviour as the number of iterations  $k$  goes to infinity. However, in practice the algorithm must terminate after finitely many iterations. There are numerous ways to interrupt CS. Some of the most popular stopping criteria include:

- i. Stop when the step size parameter  $\delta^k$  drops below a user-selected threshold, as presented in Algorithm CS.
- ii. Stop after a predefined number of function calls. This can be very useful if the evaluation of  $f$  is expensive, and a budget must not be exceeded.
- iii. Stop after a specified amount of time. For example, one might want to get the best possible value for 8 am the next morning.
- iv. Stop once the objective function value drops below a satisfactory threshold. For example, if  $f$  represents a least square function, the threshold might be a small positive value.

In practice, the context usually dictates which practical termination criteria are used. There might be more than one.

### 3.4. Selecting a Starting Point

The CS algorithm behaves very differently than the GS method. The GS method takes a wide selection of points and evaluates everything; effectively seeking a reasonable approximation of the global minimiser over the box constraint set. The CS algorithm is initiated with a single point and actively seeks improvement from that point; effectively seeking a highly accurate local minimiser. That is, the GS algorithm is a *global* optimization method, and the CS algorithm is a *local* optimization method. Being a local optimization algorithm allows CS to remove the need for the constraint set to be bounded. However, it also means that the CS algorithm requires an initial point  $x^0 \in \Omega$  from which to commence the algorithm. Clearly, the efficiency of CS is highly dependent on the initial point used.

In fact, the selection of a good initial point is an issue that will arise in most algorithms in this book. In some cases the practitioner has some professional intuition on where a good solution might be located. If this is true, then selecting an initial point is easy. However, if this is not the case, then the practitioner may wish to spend a small number, say  $p$ , of function evaluations exploring the problem and trying to determine a good initial point.

Suppose that we are willing to spend  $p$  function evaluations to explore the hyper-rectangle  $[\ell, u]$ . One strategy would be to apply GS on a grid that requires  $p$  total points. However, this strategy is not generally recommended, as the points are highly correlated one to another so the statistical distribution is not satisfactory. Indeed, observe that for any given variable  $x_i$ , there are only very few values that the variable  $x_i$  takes on the grid  $G$  of Section 3.2.

A popular, and generally more effective, technique is known as *Latin Hypercube Sampling* (LHS). This method partitions the range of each variables into  $p$  intervals. For each variable, there will be exactly one element of the sample set randomly selected in each of these  $p$  intervals. The algorithm may be written as follows.

---

**ALGORITHM 3.4. Latin hypercube sampling (LHS)**


---

Given an hyper-rectangle  $[\ell, u] \subset \mathbb{R}^n$

and  $p$  the requested number of sample points

0. **Initialisation**

let  $\Pi$  be a  $n \times p$  matrix in which each of its  $n$  rows  
is a random permutation of the row vector  $[1, 2, \dots, p]$

1. **Sample construction**

set  $x_i^j = \ell_i + \frac{(\Pi_{i,j} - r_{i,j})}{p}(u_i - \ell_i)$  for each  $i = 1, 2, \dots, n$  and  
 $j = 1, 2, \dots, p$   
and where  $r_{i,j} \in \mathbb{R}$  is randomly chosen in the interval  $[0, 1]$   
return  $\{x^1, x^2, \dots, x^p\} \subset \mathbb{R}^n$

---

Figure 3.2 shows the positions of two examples of LHS, using  $p = 10$  and 30 points in  $[0, 2] \times [0, 2] \subseteq \mathbb{R}^2$ . In each figure, there is exactly one point in each line and in each column delimited by the lines.

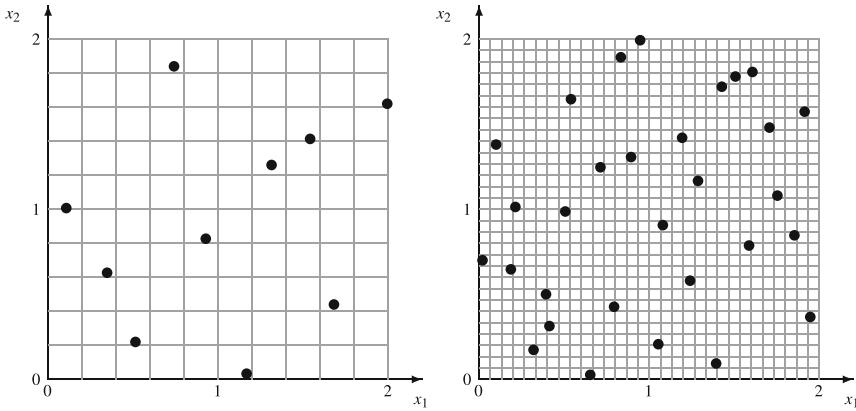


FIGURE 3.2. LHS sampling in  $\mathbb{R}^2$  with  $p = 10$  on the left and  $p = 30$  on the right

We introduce LHS now, as it can be integrated as an initial step for essentially any algorithm in this book. Moreover, several algorithms in this book allow for flexible exploratory search steps in between iterations; LHS can be employed in this regard as well.

### 3.5. Convergence Analysis of Coordinate Search

We now show that, under appropriate assumptions, the CS algorithm generates a limit point that satisfies the first order unconstrained optimality conditions discussed in Section 2.3. The structure of the proof is divided into two steps. First, it is shown that the sequence of step size parameter gets infinitely small. Second, limit points are identified, shown to exist, and shown to satisfy first order optimality conditions.

Our first proposition requires the assumption that the level sets of the objective function  $\mathcal{L}(c) := \{x \in \mathbb{R}^n : f(x) \leq c\}$  are bounded for every  $c \in \mathbb{R}$ . The purpose of this assumption is to avoid the situation where the sequence of iterates  $\{x^k\}$  is unbounded, and the minimiser is never attained.

The first result states that the step size parameter gets finer and finer.

#### PROPOSITION 3.3.

*Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be a function with bounded level sets. The sequence of step size parameters  $\{\delta^k\}$  produced by CS with  $\epsilon_{\text{stop}} = 0$  satisfies  $\lim_{k \rightarrow \infty} \delta^k = 0$ .*

**PROOF.** The proof is by contradiction. Suppose that  $\delta^k$  does not converge to zero. In this case, there must be only a finite number of unsuccessful

iterations. Let  $\ell - 1 \in \mathbb{N}$  be the index of the last unsuccessful iteration. Thus, all subsequent iterations improve the incumbent value and the step size parameter remains constant, i.e.,  $f(x^{k+1}) < f(x^k)$  and  $\delta^k = \delta^\ell$  for all  $k \geq \ell$ .

Notice that, for all  $k \geq \ell$  there exists a direction  $v^k \in \{\pm e_i : i = 1, 2, \dots, n\}$  such that  $x^{k+1} = x^k + \delta^\ell v^k$ , and therefore  $x^{k+1} = x^\ell + \delta^\ell \sum_{j=\ell}^k v_j$ . This shows that all trial points generated after iteration  $\ell$  lie on the translated integer lattice

$$M = \{x^\ell + \delta^\ell z : z \in \mathbb{Z}^n\}.$$

Since  $f(x^k) < f(x^\ell)$  for all  $k > \ell$ , we also know that all trial points generated after iteration  $\ell$  belong to the bounded level set  $\mathcal{L}(f(x^\ell))$ . The intersection of  $M$  with the bounded level set  $\mathcal{L}(f(x^\ell))$  contains finitely many points. Therefore, the sequence  $\{x^k\}_{k \geq \ell}$  cannot be infinite, a contradiction to  $\epsilon_{\text{stop}} = 0$ . Thus,  $\delta^k$  must converge to zero.  $\square$

It is worth remarking that although the term *unsuccessful* may sound negative, in fact it means that the iterate  $x^k$  has been shown to possess an objective function value which is less than or equal to those of the poll points in  $P_k$ . So, an unsuccessful iterate is actually a candidate for a local minimiser of  $f$ .

The previous proposition shows that if  $f$  has bounded level sets, then there are infinitely many unsuccessful iterations. As all iterations are contained in the level set  $\mathcal{L}(f(x^0))$ , this implies that the subsequence of unsuccessful iterates must have an accumulation point. Theorem 3.4 is our first convergence result in this book. It shows that if  $f \in \mathcal{C}^1$  and  $\hat{x}$  is any accumulation point of the subsequence of unsuccessful iterates, then  $\hat{x}$  must be a critical point for the optimization problem; i.e.,  $\nabla f(\hat{x}) = 0$ .

**THEOREM 3.4 (Convergence of CS).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be a  $\mathcal{C}^0$  function with bounded level sets and  $\{x^k\}$  be the sequence of iterates produced by CS with  $\epsilon_{\text{stop}} = 0$ . Let  $\hat{x}$  be an accumulation point of the unsuccessful iterates of  $\{x^k\}$ .

Then for any  $d \in \{\pm e_i : i = 1, 2, \dots, n\}$ , either  $f'(\hat{x}; d) \geq 0$  or  $f'(\hat{x}; d)$  does not exist. In addition, if  $f \in \mathcal{C}^1$ , then  $\hat{x}$  is a critical point:  $\nabla f(\hat{x}) = 0$ .

**PROOF.** Select any  $d \in \{\pm e_i : i = 1, 2, \dots, n\}$ . If  $f'(\hat{x}; d)$  does not exist, then we are done.

For eventual contradiction, suppose  $f'(\hat{x}; d) = -c < 0$ . Let  $\{x^{k_j}\}$  be a subsequence of unsuccessful iteration that converges to  $\hat{x}$ . Since  $f \in \mathcal{C}^1$ , there exists  $\bar{t} > 0$  and  $\varepsilon > 0$  such that

$$\frac{f(y + td) - f(y)}{t} < \frac{-c}{2} \quad \text{for all } y \in B_\varepsilon(\hat{x}), t \in (0, \bar{t}).$$

Applying  $x^{k_j} \rightarrow \hat{x}$  and  $\delta^{k_j} \rightarrow 0$ , we must have

$$\frac{f(x^{k_j} + \delta^{k_j} d) - f(x^{k_j})}{\delta^{k_j}} < \frac{-c}{2} \quad \text{for all } j \text{ sufficiently large.}$$

But, by definition of an unsuccessful iterate, and properties of  $\delta^{k_j}$ ,

$$f(x^{k_j} + \delta^{k_j} d) - f(x^{k_j}) \geq 0 \text{ and } \delta^{k_j} > 0.$$

Thus,  $f'(\hat{x}; d) \geq 0$ .

If  $f \in \mathcal{C}^1$ , then all directional derivatives are well defined and satisfy  $f'(\hat{x}; d) = \nabla f(\hat{x})^\top d$ . Thus,  $\nabla f(\hat{x})^\top e_i \geq 0$  and  $-\nabla f(\hat{x})^\top (-e_i) = \nabla f(\hat{x})^\top (-e_i) \geq 0$ . This implies  $\nabla f(\hat{x})^\top e_i = 0$  for all  $i$ , and therefore  $\nabla f(\hat{x}) = 0$ .  $\square$

Theorem 3.4 presents two sets of conclusions. The second, and stronger, conclusion is that if  $f \in \mathcal{C}^1$ , then any accumulation point of the unsuccessful iterates is a critical point. The first, and weaker conclusion is that, as long as the directional derivatives for the coordinate axes exist, then they must be nonnegative. Our next example demonstrates why this conclusion is weaker than it may appear.

**EXAMPLE 3.3.** Let  $f : \mathbb{R}^2 \mapsto \mathbb{R}$  be the convex function defined by  $f(x) = \|x\|_\infty = \max\{|x_1|, |x_2|\}$ , and let  $x^0 = [1, 1]^\top$ . The level set  $\mathcal{L}(1)$  is illustrated in Figure 3.3.

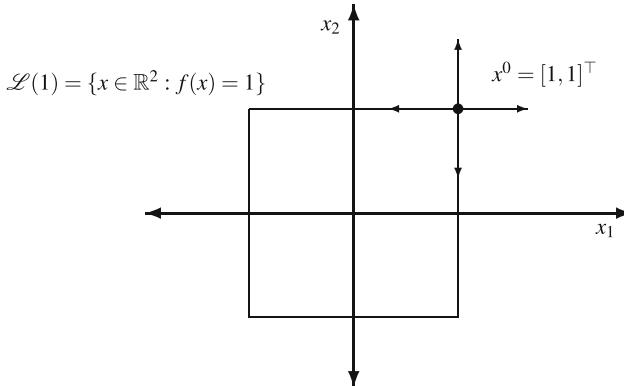


FIGURE 3.3. Applying CS on the non-differentiable convex function of Example 3.3

For any value of  $\delta > 0$ , it is not difficult to see that  $x^0 \pm \delta e_i$  (for  $i = 1$  or  $2$ ) cannot improve the incumbent value  $f(x^0) = 1$ . In particular, the directions  $e_1$  and  $e_2$  are ascent directions for  $f$ , while the directions  $-e_1$  and  $-e_2$  are parallel to the level sets of  $f$ . Thus, any implementation of CS will fail to solve this problem.

Note that this example does not violate Theorem 3.4, because the directional derivatives at the limit point  $\hat{x} = x^0$  are

$$f'(\hat{x}; e_i) = 1 \quad \text{and} \quad f'(\hat{x}; -e_i) = 0.$$


---

We will see in later chapters that the GPS and MADS algorithms possess many tools that are not present in the CS algorithm. They generalise coordinate search and allow the possibility to find good solutions for problems where the functions are not differentiable, or not continuous, or not even finite.

### 3.6. Algorithmic Variants of Coordinate Search

The poll step in the CS algorithm attempts to locate a solution by evaluating the objective function on  $2n$  points that surround the current incumbent solution. While our initial presentation did not mention it, there is some flexibility in completing this task. One simple change, for example, would be to use *opportunistic polling*, in the sense that as soon as the poll step detects a trial point  $t \in P^k$  that improves the incumbent value, then the next iterate  $x^{k+1}$  can be identified without evaluating  $f$  at the points in  $P^k$  that were not yet evaluated. The opposite of opportunist polling is referred to as *complete polling*, where all poll points are evaluated at each iteration.

If opportunistic polling is used, then the order in which the evaluations are performed will also have an effect on algorithmic performance. For example, at the beginning of an iteration, the polling directions can be *ordered* so that the directions that gave the most recent reduction in objective function values are polled first.

These three variants of the coordinate search algorithm are illustrated in the following example.

**EXAMPLE 3.4.** Consider the unconstrained minimisation of the function

$$f(x) = (5x_1 - 2)^4 + (5x_1 - 2)^2(x_2)^2 + (3x_2 + 1)^2.$$

The global minimum is at  $x^* = [\frac{2}{5}, \frac{-1}{3}]^\top$ . The CS algorithm was run three times using different polling strategies.

The first run uses complete polling: examine all four directions  $e_1, e_2, -e_1, -e_2$  and set the next iterate to be the best trial point visited up to date. The second run uses opportunistic polling: examine the four directions  $e_1, e_2, -e_1, -e_2$  in that order, but stops as soon as improvement is found. The third uses opportunistic polling with dynamic ordering: examine the four directions  $e_1, e_2, -e_1, -e_2$  and reorders them by increasing function values; it stops as soon as improvement is found.

TABLE 3.2. Three runs of the coordinate search algorithm

| Complete  |                      |        | Opportunist                            |                      |        | Ordered                                |                      |        |
|---|----------------------|--------|--|----------------------|--------|--|----------------------|--------|
| $k$   | trial point $t^\top$ | $f(t)$ | $k$                                    | trial point $t^\top$ | $f(t)$ | $k$                                    | trial point $t^\top$ | $f(t)$ |
| 0   | [3, 2]               | 29286  | 0                                      | [3, 2]               | 29286  | 0                                      | [3, 2]               | 29286  |
|   | [2, 3]               | 4772   |  | [2, 3]               | 4772   |  | [2, 3]               | 4772   |
|   | $x^1 = [1, 2]$       | 166    |  | $x^1 = [1, 2]$       | 166    |  | $x^1 = [1, 2]$       | 166    |
|   | [2, 1]               | 4176   |  |                      |        |  |                      |        |
| 1   | [2, 2]               | 4401   | 1                                      | [2, 2]               | 4401   | 1                                      | $x^2 = [0, 2]$       | 81     |
|   | [1, 3]               | 262    |  | [1, 3]               | 262    |  | [0, 3]               | 152    |
|   | $x^2 = [0, 2]$       | 81     |  | $x^2 = [0, 2]$       | 81     |  | [1, 2]               | 166    |
|   | [1, 1]               | 106    |  |                      |        |  | $x^3 = [0, 1]$       | 36     |
| 2   | [1, 2]               | 166    | 2                                      | [1, 2]               | 166    | 3                                      | $x^4 = [0, 0]$       | 17     |
|   | [0, 3]               | 152    |  | [0, 3]               | 152    |  | [0, -1]              | 24     |
|   | [-1, 2]              | 2646   |  | [-1, 2]              | 2646   |  | [0, 1]               | 36     |
|   | $x^3 = [0, 1]$       | 36     |  | $x^3 = [0, 1]$       | 36     |  | [1, 0]               | 82     |
| 3   | [1, 1]               | 106    | 3                                      | [1, 1]               | 106    | 4                                      | [-1, 0]              | 2402   |
|   | [0, 2]               | 81     |  | [0, 2]               | 81     |  | [0, -0.5]            | 17.25  |
|   | [-1, 1]              | 2466   |  | [-1, 1]              | 2466   |  | [0, 0.5]             | 23.25  |
|   | $x^4 = [0, 0]$       | 17     |  | $x^4 = [0, 0]$       | 17     |  | $x^6 = [0.5, 0]$     | 1.0625 |
| 4   | [1, 0]               | 82     | 4                                      | [1, 0]               | 82     | 5                                      | [1, 0]               | 82     |
|   | [0, 1]               | 36     |  | [0, 1]               | 36     |  | $x^7 = [0.5, -0.5]$  | 0.375  |
|   | [-1, 0]              | 2402   |  | [-1, 0]              | 2402   |  | [0.5, -1]            | 4.3125 |
|   | [0, -1]              | 24     |  | [0, -1]              | 24     |  | [1, -0.5]            | 83.5   |
| Incumbent solutions after a total of 100 function evaluations |                      |        |  |                      |        |  |                      |        |
| [0.398, -0.332] $2.2 \times 10^{-5}$                          |                      |        | [0.3999, -0.3330] $9.8 \times 10^{-7}$ |                      |        | [0.4000, -0.3334] $1.7 \times 10^{-8}$ |                      |        |

The starting point is  $x^0 = [2, 2]^\top$  with  $f(x^0) = 4401$ . Table 3.2 displays the 20 first function evaluations of these runs, as well as the incumbent solution at the 100th evaluation. The shaded boxes indicate that an improved incumbent was detected.

The order of the polling directions at iteration  $k = 0$  is  $e_1, e_2, -e_1$ , and  $-e_2$ . Under the dynamically ordered strategy, the poll directions at iterations  $k = 1$  and 2 are reordered as  $-e_1, e_2, e_1$ , and  $-e_2$  because  $f(x^0 - e_1) = 166 < f(x^0 + e_2) = 4772 < f(x^0 + e_1) = 29286$  and because  $f(x^0 - e_2)$  was not evaluated.

---

Examining Table 3.2, we notice that several redundant function evaluations are performed. For example, at iteration  $k = 3$  with the *complete* strategy, the two first poll points  $[1, 1]^\top$  and  $[0, 2]^\top$  were previously visited at iteration  $k = 1$ . Evaluating the objective function at these points is a waste of resources, especially if the functions are expensive to evaluate. This redundancy is not surprising since the frame centre  $x^k$  of a successful iteration necessarily belongs to the next frame  $P^{k+1}$ . A way to circumvent this redundant work is to store in a cache each trial point together with its objective function value. Then, each time a trial point is generated, the cache is consulted to see if that point was previously generated.

Let us compare the various implementations of the CS algorithm on the two variants of the rheology parameter fit optimization problem from Section 1.3.3. Simultaneously, we will compare with the solutions generated with the GS algorithm from Example 3.1.

**EXAMPLE 3.5.** Recall the rheology parameter fit optimization problem from Section 1.3.3. In Example 3.1 we rescaled this problem to create the pair of objective functions  $\hat{f}(x) = \hat{g}(520x_1, 14x_2, 0.038x_3)$  and  $\check{f}(x) = \check{g}(520x_1, 14x_2, 0.038x_3)$ , to be minimised over the box  $x \in [0, 20]^3$ . In this example, we use a budget of 125 function evaluations to find a good initial point, and then apply the CS algorithm to find a minimiser. For simplicity, to select initial points we only work with  $\hat{f}$ .

We have two methods for selecting our initial point, GS and LHS. The GS method creates a  $5 \times 5 \times 5$  grid of points:  $x_i \in \{0, 5, 10, 15, 20\}$  for each  $i$ . Working with  $\hat{f}$ , the initial point resulting from GS is  $x_{\text{GS}}^0 := [15, 20, 10]^\top$  with  $\hat{f}(x_{\text{GS}}^0) = 422.506$ .

The LHS method involves some random number generation, so we repeat the experiment 6 times in order to better understand the expected behaviour. The LHS method generated 125 trial points, and kept the best one. In our tests, working with  $\hat{f}$ , we ended up creating initial points

$$\begin{aligned} x_{\text{LHS}_1}^0 &= [8.172517606, \quad 5.058263716, \quad 5.444856567]^\top \quad \hat{f}(x_{\text{LHS}_1}^0) = 692.324 \\ x_{\text{LHS}_2}^0 &= [13.04832254, \quad 15.84400309, \quad 9.950620587]^\top \quad \hat{f}(x_{\text{LHS}_2}^0) = 379.782 \\ x_{\text{LHS}_3}^0 &= [12.31453665, \quad 13.75028434, \quad 9.557207957]^\top \quad \hat{f}(x_{\text{LHS}_3}^0) = 293.003 \\ x_{\text{LHS}_4}^0 &= [11.36633281, \quad 12.12935162, \quad 8.906909739]^\top \quad \hat{f}(x_{\text{LHS}_4}^0) = 309.718 \\ x_{\text{LHS}_5}^0 &= [9.690281657, \quad 6.799833301, \quad 5.904578444]^\top \quad \hat{f}(x_{\text{LHS}_5}^0) = 825.650 \\ x_{\text{LHS}_6}^0 &= [12.20082785, \quad 12.61627174, \quad 8.890182552]^\top \quad \hat{f}(x_{\text{LHS}_6}^0) = 188.710. \end{aligned}$$

We then ran the CS algorithm, using the complete polling, opportunistic polling, and opportunistic polling with dynamic ordering, using 125 additional function calls and 375 additional function calls, for a total of 250 and 500 function calls, respectively. The results appear in Table 3.3 for non-smooth objective function  $\hat{f}$  and Table 3.4 for smooth objective function  $\check{f}$ .

In Tables 3.3 and 3.4 we see several trends. First of all, with just one exception, the GS strategy to find an initial point produces a worst value than every LHS initialisation. Second, the randomness inherent to LHS greatly affects the final results. Notice that the overall best solutions were generated by the sampling leading to the worst objective function value:  $\hat{f}(x_{\text{LHS}_5}^0) = 825.650$ . Third, increasing the number of additional function evaluations from 125 to 375 never deteriorates, but does not always improve the final value. The largest improvement is with  $x_{\text{LHS}_1}^0$  where the objective function value dropped by almost 37 for  $\hat{f}$  with the opportunistic strategy, and dropped almost 6000 for  $\check{f}$  with the complete strategy. Fourth, the complete strategy is very seldom preferable to the other two strategies. Similarly, the ordered strategy tends to outperform the opportunistic without ordered

TABLE 3.3. Best objective function value  $\hat{f}$  of the non-smooth rheology problem, generated using different initial points and different variants of the CS algorithm

| Initial Point | Complete             |                      | Opportunistic        |                      | Ordered              |                      |
|---------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
|               | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls |
| $x_{GS}^0$    | 299.884              | 299.883              | 299.883              | 299.883              | 299.883              | 299.883              |
| $x_{LHS_1}^0$ | 206.311              | 201.347              | 150.556              | 113.846              | 200.930              | 199.190              |
| $x_{LHS_2}^0$ | 230.626              | 230.611              | 230.613              | 230.611              | 220.553              | 220.546              |
| $x_{LHS_3}^0$ | 217.023              | 217.004              | 217.023              | 217.004              | 215.069              | 215.069              |
| $x_{LHS_4}^0$ | 138.496              | 138.489              | 138.493              | 138.490              | 136.042              | 136.040              |
| $x_{LHS_5}^0$ | 35.020               | 34.428               | 46.178               | 46.169               | 46.132               | 46.129               |
| $x_{LHS_6}^0$ | 170.123              | 170.121              | 170.121              | 170.121              | 170.080              | 170.080              |

TABLE 3.4. Best objective function value  $\check{f}$  of the smooth rheology problem, generated using different initial points and different variants of the CS algorithm

| Initial Point | Complete               |                        | Opportunistic          |                        | Ordered                |                        |
|---------------|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
|               | 125 $\check{f}$ -calls | 375 $\check{f}$ -calls | 125 $\check{f}$ -calls | 375 $\check{f}$ -calls | 125 $\check{f}$ -calls | 375 $\check{f}$ -calls |
| $x_{GS}^0$    | 8241.46                | 8182.19                | 8241.46                | 8186.61                | 8033.51                | 7878.70                |
| $x_{LHS_1}^0$ | 10147.10               | 4168.96                | 5139.76                | 1094.98                | 5150.02                | 1572.92                |
| $x_{LHS_2}^0$ | 6281.41                | 6162.82                | 6272.51                | 6002.77                | 5819.71                | 5310.60                |
| $x_{LHS_3}^0$ | 4732.67                | 4481.86                | 4734.38                | 4522.74                | 4657.93                | 4325.62                |
| $x_{LHS_4}^0$ | 3752.66                | 3457.25                | 3925.46                | 3714.71                | 3856.69                | 3218.25                |
| $x_{LHS_5}^0$ | 1145.47                | 738.55                 | 257.87                 | 237.01                 | 247.21                 | 217.62                 |
| $x_{LHS_6}^0$ | 3691.67                | 3382.16                | 4534.16                | 4309.76                | 4458.49                | 3838.81                |

polling. However, in both cases exceptions occur, so this cannot be taken as a universal rule.

Comparing to Example 3.1, all CS variants yield a better final value than GS with 512 points, except those working with the nonsmooth function  $\hat{f}$  initiated at  $x_{GS}^0$ . This suggests that considerable saving in computational resources can be achieved even with an elementary optimization algorithm.

### 3.7. Methods in This Book

In Example 3.5, we saw how even a basic DFO algorithm can improve on naive methods. Moving forward in this book, we shall examine six more advanced algorithms for solving BBO problems. Table 3.5 gives a high-level view of these methods and provides the main algorithmic idea behind them.

TABLE 3.5. A summary of the optimization methods covered in this book

| Chapter and Method                    | Type                                    | Idea   |
|---------------------------------------|---|--|
| 4. Genetic Algorithm (GA)             | Global search heuristic of individuals. | Examine a “population” of potential solutions, breed new potential solutions based on “fitness”  |
| 5. Nelder-Mead (NM)                   | Local search heuristic                  | Evaluate and rank the vertices of a simplex (see Section 2.6). Replace worst one by reflecting, expanding, or contracting the simplex.                   |
| 7. Generalised pattern search (GPS)   | Direct-search optimization for BBO      | Advances on CS to allow for a greater variety of “pattern” directions, and allow for explorative subroutines.  |
| 8. Mesh adaptive direct search (MADS) | Direct-search optimization for BBO      | Advances on GPS to allow for an infinite set of exploratory directions, and deal with general inequality constraints.                                    |
| 10. Model-based descent method (MBD)  | Model-based optimization for DFO        | Use approximate gradients to predict descent directions, then apply a line search to seek a minimiser.   |
| 11. Model-based trust region (MBTR)   | Model-based optimization for DFO        | Minimise a model of the objective function over a trust region and evaluate the candidate point, then either update the model or the incumbent solution. |

The first two methods are covered in Part 2, the next two are presented in Part 3, and the final two are detailed in Part 4. Finally, Part 5 shows how to extend and combine some of the ideas and principles presented throughout the book.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 3.1.** + Consider  $\min_x \{f(x) : x \in \Omega\}$  where  $f(x) = -|x|$  and  $\Omega = (-1, 1]$ .

a) Which hypotheses of Theorem 3.1 are satisfied?

b) Let  $S = \{-1/2, 0, 1/2,$

$$\begin{aligned} & -3/4, -2/4, -1/4, 0, 1/4, 2/4, 3/4, \\ & -7/8, -6/8, \dots, 6/8, 7/8, \\ & \dots \}. \end{aligned}$$

Prove that  $S$  is dense in  $\Omega$ .

c) Show that applying ES using  $S$  (with this order) generates a sequence of iterates that fails to satisfy

$$\liminf_{k \rightarrow \infty} \text{dist}(x_{\text{best}}^k, X^*) = 0.$$

Explain why this does not contradict Theorem 3.1.

**EXERCISE 3.2.**  $\square$  In this question we explore why compactness is required to guarantee that the sequence  $x_{\text{best}}^k$  from ES converges to a minimiser. Consider the single-variable function  $f(x) = -e^{-x^2} + e^{x+1/4}$ .

- a) Find  $\min_x \{f(x) : x \leq 0\}$  and  $\operatorname{argmin}_x \{f(x) : x \leq 0\}$  (analytically prove your result).

- b) The following is a countable dense sequence in  $\{x : x \leq 0\}$

$$\begin{aligned} S_1 = \{0, -1/2, -2/2, -1/4, -2/4, \dots, -8/4, \\ -1/8, -2/8, \dots, -32/8, \\ -1/16, -2/16, \dots, -128/16, \\ \dots\}. \end{aligned}$$

Analytically determine what ES will do if this sequence (with this order) is used.

- c) We can adjust the sequence in part b) as follows:

any time the value  $-1/2$  arises, replace it with  $-10^j$ ,

where  $j$  is incremented by 1 each time a replacement is made. e.g.,

$$S_2 = \{0, -10^1, -2/2, -1/4, -10^2, -3/4, \dots\}.$$

Numerically determine what ES will do if this sequence (with this order) is used.

**EXERCISE 3.3.** Assume the notation and conditions of Theorem 3.2.

- a) Select any  $x^* \in \operatorname{argmin}_x \{f(x) : x \in [\ell, u]\}$ . Let  $p^* \in \operatorname{argmin}_x \{\|p - x^*\| : p \in G\}$  (i.e.,  $p^*$  is the closest point in  $G$  to  $x^*$ ). Prove  $\|x^* - p^*\| \leq \sqrt{n}\frac{\delta}{2}$ .
- b) Prove  $|f^* - f(p^*)| \leq K\sqrt{n}\frac{\delta}{2}$  and complete the proof of Theorem 3.2.

**EXERCISE 3.4.** Suppose  $f : \mathbb{R}^n \mapsto \mathbb{R}$  has a Lipschitz constant 1, and GS is used to solve

$$f^* = \min_x \{f(x) : x \in [0, 1]^n\}.$$

Suppose a solution is desired such that

$$f_{best} - f^* \leq 10^{-3}.$$

- a) If  $n = 1$ , how many function evaluations will be required?
- b) If  $n = 2$ , how many function evaluations will be required?
- c) More generally, how many function evaluations will be required as a function of  $n$ ?
- d) If a function evaluation takes 1 millisecond ( $10^{-3}$  seconds), and you want to get your solution before the sun explodes (in approximately 2.8 billion years), what is the maximum dimension you can solve?

**EXERCISE 3.5.** Explain how it is possible that using  $p = 10$  produces a better result than  $p = 22$  in Table 3.1.

**EXERCISE 3.6.** Apply CS to the minimisation of the continuously differentiable function  $f(a, b) = -(ab)^2$  from the initial point  $x^0 = [0, 0]^\top$ .

- a) Prove that the algorithm never moves from the origin.
- b) Explain why this does not contradict Theorem 3.4.

- c) From a different initial point, would it be possible for CS to generate iterates that converge to the origin? (Justify your conclusion.)

**EXERCISE 3.7.** + Apply the three variants of the CS algorithm (complete polling, opportunistic polling, and opportunist with dynamic ordering) to the continuously differentiable function  $f(x_1, x_2) = e^{-x_1} + x_2$  from  $x^0 = [0, 0]^\top$  with initial poll ordering  $e_1, e_2, -e_1$  and  $-e_2$ . Explain the results.

**EXERCISE 3.8.**  $\square$  Consider the function

$$f(x_1, x_2) = \begin{cases} ((x_1)^2 + 1)(x_2)^2 & \text{if } x_2 \geq 0, \\ ((1 - x_1)^2 + 1)(x_2)^2 & \text{if } x_2 < 0. \end{cases}$$

- a) Prove that  $f$  is continuously differentiable ( $\mathcal{C}^1$ ) on  $\mathbb{R}^2$ .
- b) Plot some level sets of  $f$ .
- c) Show that CS with complete polling,  $x^0 = [0, \frac{2}{3}]^\top$ , and  $\delta^0 = 1$  generates infinitely many accumulation points.  
[Hint: Using a computer to launch CS might help you to see a pattern.]
- d) What is  $\nabla f$  at the accumulation points?

**EXERCISE 3.9.** Table 3.2 displays the 20 first trial points generated by the three coordinate search variants: complete, opportunistic, and ordered on the objective function

$$f(a, b) = (5a - 2)^4 + (5a - 2)^2b^2 + (3b + 1)^2.$$

Show that exactly two of these variants generate the same 21<sup>st</sup> trial point.

**EXERCISE 3.10.** + This exercise proposes to create and use a surrogate function to order the poll points following an unsuccessful iteration of CS.

- a) Suppose that iteration  $k$  of the CS algorithm was unsuccessful. Show how to use centred differences to estimate the gradient of  $f$  at  $x^k$ .
- b) Use the estimation in a) to create a linear approximation  $s(x)$  of the function  $f(x)$ .
- c) Illustrate a) and b) on iteration  $k = 4$  of Example 3.4 with complete polling.
- d) How would you use the surrogate function  $s(x)$  to order the poll points of iteration  $k + 1$ ? (Combined with the opportunistic strategy, ordering the poll points can reduce the number of calls to the function  $f$ ). Illustrate on iteration  $k + 1 = 5$  of Example 3.4.

**EXERCISE 3.11.** Let  $f : \mathbb{R}^2 \mapsto \mathbb{R}$  be the convex function defined by  $f(x) = \|x\|_1 = |x_1| + |x_2|$ .

- a) Draw level sets of  $f$ .
- b) Suppose  $x^0 \neq [0, 0]^\top$ . Prove that, if  $\delta$  is sufficiently small, then there exists  $i$  such that either  $f(x^0 + \delta e_i) < f(x^0)$  or  $f(x^0 - \delta e_i) < f(x^0)$ .
- c) Prove that if CS is applied to minimise  $f$ , then the algorithm will be successful regardless of the starting point.  
[Hint: do not forget to consider  $x^0 = [0, 0]^\top$ .]

**EXERCISE 3.12.** + Consider an unconstrained optimization problem where all variables are constrained to be integers (positive or negative). Propose a modification of CS to handle such a problem, including a pertinent stopping criteria.

**EXERCISE 3.13.** □ Implement CS using complete polling, opportunistic polling, and opportunistic polling with dynamic ordering. Recreate Example 3.5 using your implementation. Explain how (and why) your results differ from those in Tables 3.3 and 3.4.

□

**Chapter 3 Project: Tuning Runge-Kutta Parameters.** This project studies a question frequently encountered in engineering, mathematics, and computer science: tuning algorithmic parameters. In this project, the question of finding good parameter values for the well-known Runge-Kutta algorithm is formulated as a blackbox optimization problem.

The fourth-order Runge-Kutta method solves a system of  $\ell$  ordinary differential equations of the type  $y'(t) = f(t, y(t))$  subject to the initial condition  $y(t^0) = y^0 \in \mathbb{R}^\ell$  with  $t^0 \in \mathbb{R}$ , where  $y : \mathbb{R} \mapsto \mathbb{R}^\ell$  and  $f : \mathbb{R} \times \mathbb{R}^\ell \mapsto \mathbb{R}^\ell$ . Runge-Kutta uses a fixed step size  $h > 0$  and estimates  $y(t^0 + nh)$  by  $y^n$  using the following iterative process:

$$\begin{aligned} k_1 &= hf(t^n, y^n) \\ k_2 &= hf(t^n + \beta_1 h, y^n + \beta_1 k_1) \\ k_3 &= hf(t^n + \beta_2 h, y^n + \beta_3 k_2 + (\beta_2 - \beta_3)k_1) \\ k_4 &= hf(t^n + \beta_4 h, y^n + \beta_5 k_2 + \beta_6 k_3 + (\beta_4 - \beta_5 - \beta_6)k_1) \\ y^{n+1} &= y^n + \alpha_1 k_1 + \alpha_2 k_2 + \alpha_3 k_3 + \alpha_4 k_4 \\ n &\leftarrow n + 1. \end{aligned}$$

The values  $\alpha = [\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}]^\top$  and  $\beta = [\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 0, 1]^\top$  correspond to the classical Runge-Kutta method, but any set of parameters  $(\alpha, \beta) \in \mathbb{R}^4 \times \mathbb{R}^6$  that satisfy the following eight nonlinear equations define a valid method:

$$\begin{aligned} \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 &= 1, & \alpha_3 \beta_1 \beta_3 + \alpha_4 \beta_1 \beta_5 + \alpha_4 \beta_2 \beta_6 &= \frac{1}{6}, \\ \alpha_2 \beta_1 + \alpha_3 \beta_2 + \alpha_4 \beta_4 &= \frac{1}{2}, & \alpha_3 \beta_1 \beta_2 \beta_3 + \alpha_4 \beta_1 \beta_4 \beta_5 + \alpha_4 \beta_2 \beta_4 \beta_6 &= \frac{1}{8}, \\ \alpha_2 \beta_1^2 + \alpha_3 \beta_2^2 + \alpha_4 \beta_4^2 &= \frac{1}{3}, & \alpha_3 \beta_1^2 \beta_3 + \alpha_4 \beta_1^2 \beta_5 + \alpha_4 \beta_2^2 \beta_6 &= \frac{1}{12}, \\ \alpha_2 \beta_1^3 + \alpha_3 \beta_2^3 + \alpha_4 \beta_4^3 &= \frac{1}{4}, & \alpha_4 \beta_1 \beta_3 \beta_6 &= \frac{1}{24}. \end{aligned}$$

These equations have two degrees of freedom, and given values of  $\beta_1$  and  $\beta_5$ , the other parameters can be identified according to the following equations:

$$\beta_2 = \begin{cases} \frac{1}{2} & \text{if } \beta_1 = \frac{1}{2} \\ \frac{12\beta_1^3\beta_5 - 6\beta_1\beta_5 - \sqrt{\gamma} - 5\beta_1 + 5}{8(3\beta_1^2\beta_5 - 2\beta_1\beta_5 - \beta_1 + 1)} & \text{otherwise} \end{cases}$$

where  $\gamma = 144\beta_1^6\beta_5^2 - 384\beta_1^5\beta_5^2 + 400\beta_1^4\beta_5^2 - 40\beta_1^4\beta_5 - 192\beta_1^3\beta_5^2 + 72\beta_1^3\beta_5 + 36\beta_1^2\beta_5^2 + 16\beta_1^3 - 36\beta_1^2\beta_5 - 39\beta_1^2 + 4\beta_1\beta_5 + 30\beta_1 - 7;$

$$\beta_3 = \begin{cases} \frac{1}{2(1 - \beta_5)} & \text{if } \beta_1 = \frac{1}{2} \\ \frac{\beta_2(\beta_1 - \beta_2)}{2\beta_1(2\beta_1 - 1)} & \text{otherwise;} \end{cases}$$

$$\beta_4 = 1;$$

$$\beta_6 = \frac{\beta_1(-\beta_1\beta_2\beta_5 + \beta_2^2\beta_5 + \beta_1\beta_3 - \beta_3)}{12\beta_1^3\beta_3^2 - 4\beta_1^2\beta_2\beta_3 - 8\beta_1^2\beta_3^2 + 4\beta_1\beta_2^2\beta_3 + \beta_1\beta_2^2 - \beta_3^2};$$

$$\alpha_2 = \frac{12\beta_1^2\beta_2^2\beta_3\beta_5 - 8\beta_1^2\beta_2\beta_3\beta_5 - 4\beta_1^2\beta_3^2 - 4\beta_1\beta_2^2\beta_3 + 4\beta_1\beta_2\beta_3 + \beta_2^3 - \beta_2^2}{24\beta_1^3\beta_3(-\beta_1\beta_2\beta_5 + \beta_2^2\beta_5 + \beta_1\beta_3 - \beta_3)},$$

$$\alpha_3 = \frac{12\beta_1^3\beta_3\beta_5 - 8\beta_1^2\beta_3\beta_5 - 4\beta_1^2\beta_3 + \beta_1\beta_2 + 4\beta_1\beta_3 - \beta_2}{24\beta_1^2\beta_3(\beta_1\beta_2\beta_5 - \beta_2^2\beta_5 - \beta_1\beta_3 + \beta_3)};$$

$$\alpha_4 = \frac{12\beta_1^3\beta_3^2 - 4\beta_1^2\beta_2\beta_3 - 8\beta_1^2\beta_3^2 + 4\beta_1\beta_2^2\beta_3 + \beta_1\beta_2^2 - \beta_3^2}{24\beta_1^2\beta_3(-\beta_1\beta_2\beta_5 + \beta_2^2\beta_5 + \beta_1\beta_3 - \beta_3)};$$

$$\alpha_1 = 1 - \alpha_2 - \alpha_3 - \alpha_4.$$

Given the 2 degrees of freedom, some researchers have proposed alternatives to the Runge-Kutta method. For example, Gill's method uses  $\beta_1 = \frac{1}{2}$  and  $\beta_5 = \frac{-1}{\sqrt{2}}$ , and Ralston's method uses  $\beta_1 = \frac{2}{5}$  and  $\beta_5 = -\frac{1523\sqrt{5}}{1276} - \frac{975}{2552}$ .

In this project we seek to identify *good* values of these parameters in order to solve the following family of  $\ell \geq 2$  differential equations,

$$y'_i(t) = \frac{y_i^2}{t(1+i)} - (1+i) \sqrt{\frac{(i+1)(2+i+y_{i+1})}{2+i-y_{i+1}}}, \quad i = 1, 2, \dots, \ell-1$$

$$y'_\ell(t) = \frac{y_\ell^2}{t(1+\ell)} - (1+\ell) \sqrt{\frac{2+y_1}{2-y_1}},$$

with  $y_i(1) = 1 - i, \quad i = 1, 2, \dots, \ell.$

We wish to tune the parameters  $\beta_1$  and  $\beta_5$  on systems of this family with  $\ell \in \{4, 5, 6, 7\}$  equations to estimate the value of  $y(4) \in \mathbb{R}^\ell$  using a number of steps  $n$  ranging between 145 and 150.

- a) For given values of  $\ell, n, \beta_1$ , and  $\beta_5$ , propose a way to quantify the quality of the solution produced by the Runge-Kutta method.

You will need to compare the solution with an approximation of the exact solution  $y(4)$  that you will have determined using a smaller step size  $h$ . Discuss the choice of norms.

- b) Formulate the question of identifying good parameter values as an unconstrained BBO problem whose objective function takes as input the values  $\beta_1$  and  $\beta_5$ .
- c) Using CS to propose good values for  $\beta_1$  and  $\beta_5$ .

# *Some Remarks on DFO*

---

Each part of this book concludes with some remarks pertinent to the topic just covered. These notes provide some contextualisation with respect to the existing scientific literature, both historical and current, and citations for specific details that were omitted.

In regards to both historical and current scientific literature in derivative-free optimization, it would be remiss of us to not begin by mentioning the excellent textbook of Conn, Scheinberg and Vicente [47] titled “Introduction to Derivative Free Optimization”. Their textbook provided a major boost to the field of DFO and BBO, won the Lagrange Prize in Continuous Optimization in 2015, and remains an excellent source for DFO theory. Many technical details that are omitted in the present book can be found in that source. Conversely, much of the background material and basic steps that are omitted in that textbook can be found here.

The material in Chapter 1 is largely a result of our personal experiences in the field of DFO and BBO. In particular, we should note that our definitions of DFO and BBO represent our opinion on the language and is not universally standardised.

More information about the three applications presented in Section 1.3 may be found in the following sources. The tsunami-detection buoys positioning problem is described in [155]. The earthquake retrofitting problem is described in [29]. The data for the parameter fit rheology problem comes from [38], and the optimization problem formulation comes from the numerical analysis textbook [76]. Some other examples of parameter fitting problems solved via DFO include queueing [26], forecasting [90], algorithmic [23] and chemistry [96].

Numerous other examples of successful applications of DFO algorithms exist. A large collection of BBO/DFO applications in engineering, along with some of the algorithms used, can be found in the surveys [10, 54, 95].

The background in Chapter 2 is now fairly classical in the field of Mathematics. Textbooks on matrix algebra and multivariate calculus can be found

in great abundance. The volume of a simplex presented here coincides with the formal definition of volumes of objects in  $\mathbb{R}^n$  [156]. The variational analysis material (sets, convexity, and simplices) is less common, but there are nonetheless a number of great sources: for example, the “bible” on convexity by Rockafellar [150].

In Chapter 3, we presented a brute force GS method. While simple to implement, this method should never be used in practice, as was discussed at the time.

The basic CS algorithm dates back to at least 1952, when a variant was suggested by Fermi and Metropolis [68]. Fermi and Metropolis did not view it as a competitive optimization algorithm, only a demonstration of the new power to use the Maniac computer in the Los Alamos National Laboratory to automate minimisation. The same procedure was used in 1955 at the Argonne National Laboratory in a six parameter pion-proton data fitting problem [7]. The CS algorithm is also known as *compass search*, and an historical presentation can be found in the survey of Kolda, Lewis, and Torczon [109].

The final project on parameter tuning [23, 25, 100] from the previous chapter was studied in [11]. The Gill parameters [83] minimise storage requirements by reusing function evaluations and the Ralston [143] parameters minimise the theoretical truncation error, i.e., the coefficient multiplying the  $h^4$  error term.

PART

# 2

## Popular Heuristic Methods

Part 2 examines some heuristic optimization methods frequently used in the literature.

- Chapter 4 describes genetic algorithms. These methods are popular, in part because of the nice metaphor that they invoke. However, they fail to be supported by a useful convergence analysis.
- Chapter 5 presents the widely popular method of Nelder and Mead. This local method can be surprisingly efficient on many unconstrained optimization problems; however, it can also be shown to fail on a low-dimension smooth convex problem.

Conclusion: while heuristics can be useful in many applications, they cannot be relied on to guarantee convergence.

# *Genetic Algorithms*

Optimization algorithms for blackbox functions can be broadly split into two categories: heuristic and non-heuristic. A *heuristic* is any approach that, while supported by some argument of why it should succeed, does not include a guarantee of success. In the framework of blackbox optimization, we take this statement to mean that the algorithm does not employ mathematically proven stopping criterion. Hence, derivative-free optimization is distinguished from heuristic blackbox optimization, by the presence of a stopping test that provides some assurance of optimality.

The majority of this book focuses on non-heuristic optimization methods for blackbox functions. However, to truly appreciate what sets DFO apart from heuristic approaches, it is valuable to explore at least a few of the more popular heuristic methods. Before continuing to genetic algorithms, it is worth pointing out that while heuristic methods may not have the mathematical structure of a DFO method, they may be nonetheless fairly effective in practice. They are generally easy to understand, easy to implement, and (as a result) more widely used than DFO methods. For one-off problems where function evaluations are very quick, it may be more reasonable to use a quickly implemented heuristic method, than to spend a long time implementing a top-of-the-line DFO method.

Another reason we explore heuristic methods, and in particular genetic algorithms, is that most of the DFO methods discussed in this book will allow for imbedded heuristic subroutines. This is important as the DFO methods are all locally convergent methods, meaning they will converge to a critical point, but not necessarily the global minimiser of a problem. Embedding global search heuristics is a natural way to allow DFO methods to break free of local minimisers. This is discussed further in Parts 3 and 4 where some DFO methods are examined. For now, it is important to stress that genetic algorithms are global search heuristics, and so, they place their emphasis on finding a decent global solution to a problem, as opposed to finding a highly accurate local solution to a problem.

### 4.1. Biology Overview and the GA Algorithm

Genetic algorithms, which are also called evolutionary strategies, are algorithms that use “survival-of-the-fittest” to weed out poor solutions and breed new solutions to an optimization problem. In keeping with the biological theme of the method, points are often referred to as individuals and a collection of points forms a population. Information within a point is *encoded* as *chromosomes*, and biological processes such as *crossover* and *mutation* are used to create new individuals.

**ALGORITHM 4.1.** Genetic algorithm (GA)

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and an initial population  $P^0 = \{x^1, x^2, \dots, x^{\bar{p}}\}$

0. Initialise:

|                     |                      |
|---------------------|----------------------|
| $\gamma \in (0, 1)$ | mutation probability |
| $k \leftarrow 0$    | iteration counter    |

1. Fitness:

| Use  $f(x)$  to assign a fitness to each individual  $x \in P^k$

2. Reproduce:

2a) Selection: select 2 parents from  $P^k$  and proceed to 2b)

or select 1 survivor from  $P^k$  and proceed to 2d)

2b) Crossover: use the 2 parents to create an offspring

2c) Mutation: with probability  $\gamma$  mutate the offspring  
(with probability  $1 - \gamma$  the offspring is not mutated)

check the feasibility of the mutated offspring

if the offspring is infeasible

declare the offspring deceased and return to 2a)

otherwise declare the offspring a survivor and proceed to 2d)

2d) Update next generation: place survivor into  $P^{k+1}$

if  $|P^{k+1}| \geq \bar{p}$  declare population complete and proceed to 3

otherwise declare population incomplete and go to 2a)

---

 3. Update:

| increment  $k \leftarrow k + 1$ , stop or go to 1

---

Examining GA, we notice that the method leaves a great deal of flexibility in interpretation. One primary example is at Step 3, where no guidance is given on how to make the decision on whether to loop or stop. This is one of the reasons that GA is classified as a heuristic method, and not a DFO algorithm.

We see similar flexibility in:

- i. The input initial population, where no guidance is given on how to build  $P^0$ .
- ii. Step 1, where  $f(x)$  is used to define a new metric called fitness.
- iii. Step 2a), where no guidance is given on how to decide between selecting 2 parents or 1 survivor, or how to select the individual(s).
- iv. Step 2b), where we use two parents to create a new offspring.
- v. Step 2c), where the offspring is mutated and possibly declared deceased.

Some of this flexibility will be removed in later sections of this chapter. However, for the most part, later sections will only provide examples of how to make some of these decisions, and not prescribe a detailed algorithm.

This flexibility is one of the strengths and weaknesses of GA. On the negative side, it means that no two genetic algorithms should be expected to behave the same, even if the optimization problem is identical. On the positive side, it allows the implementor to apply some level of customisation to create a method that works for their particular problem.

## 4.2. Fitness and Selection

In order to stay true to the evolutionary theme, GA refers to the quality of a point as its *fitness*. While the fitness value for a point  $x$  should be highly related to the function value  $f(x)$ , assigning fitness has some flexibility incorporated in it as well. There are two key rules when assigning fitness values to a point.

- i. When seeking a minimiser of  $f$ , if  $f(x) < f(y)$  then the fitness value of  $x$  should be greater than that of  $y$  (this rule is reversed in a maximisation context).
- ii. The fitness value of a point should be strictly positive.

The reason for the first rule should be obvious. The second rule is required for some of the selection and crossover techniques discussed shortly. We end this section by providing two example methods of assigning fitness.

**DEFINITION 4.1 (Rank Fitness).**

Given a population  $P^k = \{x^1, x^2, \dots, x^{\bar{p}}\}$ , and corresponding function values  $\{f(x^1), f(x^2), \dots, f(x^{\bar{p}})\}$ . The rank fitness of a point  $x^i$  is the number  $m$  of points  $x^j$  ( $j \in \{1, 2, \dots, m\}$ ) where  $f(x^i)$  is less than or equal to  $f(x^j)$ .

Note that, when determining rank fitness of  $x^i$ , one must include it in the list of points to rank against, otherwise the worst point might have a fitness of 0 (whereas, we need fitness to be strictly positive).

Rank fitness is easy and quick. In rank fitness once the fitness is determined, the function value is immediately discarded. While this may be advantageous in some (rare) situations, for an optimization algorithm it is a little peculiar. An alternative to rank fitness is function valued fitness.

**DEFINITION 4.2 (Function Valued Fitness).**

Given a population  $P^k = \{x^1, x^2, \dots, x^{\bar{p}}\}$ , the function valued fitness is obtained by multiplying the function value by  $-1$  and then shifting the results until the lowest value is equal to 1. This creates the following formula

$$\text{Fit}(x^i) = -f(x^i) + \bar{f} + 1 \quad \text{where } \bar{f} = \max_{i=1,2,\dots,\bar{p}} \{f(x^i)\}.$$

**EXAMPLE 4.1.** Suppose  $f(x) = \sin(\pi x)$ , and  $P^k = \{-0.1, 0.1, 0.3, 0.7\}$ . The unordered set of function values is (approximately)  $\{-0.309, 0.309, 0.809, 0.809\}$ . The rank fitness of  $x^2 = 0.1$  is 3, and the function value fitness of  $x^2$  is obtained by first computing  $\bar{f} \approx 0.809$ , and then setting  $\text{Fit}(x^2) \approx -0.309 + 0.809 + 1 = 1.500$ .

---

Now that fitness is established, we enter the selection stage (Step 2a)). In this stage, we aim to select either 1 survivor (which will be immediately placed into  $P^{k+1}$ ), or 2 parents (which will be used to create an offspring). Selecting 2 parents is essentially the same as selecting a single survivor two times – possibly with the restriction that the parents cannot be the same individual. Some popular selection strategies include *elitism*, *roulette wheel*, and *tournament selection*.

**DEFINITION 4.3 (Elitism Selection).**

In order to ensure the best known solution is not lost, the elitism strategy simply selects the individual with the highest fitness level (with ties broken by random selection with uniform probability). If elitism is used multiple times, say to select a total of  $s$  survivors, then the  $s$  individuals with the highest fitness levels are selected.

Elitism is the simplistic strategy, but is also an extremely important strategy. In particular, we shall see that in order to ensure convergence, it is important to select at least one survivor using the elitism strategy.

**DEFINITION 4.4 (Roulette Wheel Selection).**

Given individuals  $\{x^1, x^2, \dots, x^{\bar{p}}\}$  with corresponding fitness values  $\{f^1, f^2, \dots, f^{\bar{p}}\}$ , roulette wheel selection randomly selects individuals with probability

$$\text{prob}(x^i) = f^i / \sum_{j=1}^{\bar{p}} f^j.$$

Notice, as  $f^i > 0$  for all individuals, roulette selection provides  $\text{prob}(x^i) > 0$  for all individuals. Also note that, in roulette selection,

$$\sum_{i=1}^{\bar{p}} \text{prob}(x^i) = \sum_{i=1}^{\bar{p}} f^i / \sum_{j=1}^{\bar{p}} f^j = 1,$$

resulting in a valid probability distribution function.

**EXAMPLE 4.2.** Suppose individuals  $\{x^1, x^2, x^3, x^4, x^5\}$  have corresponding fitness values

$$\{f^1, f^2, f^3, f^4, f^5\} = \{0.5, 1, 4, 1.5, 3\}.$$

If the elitism strategy is used to select 2 survivors, then those survivors will be  $x^3$  and  $x^5$ . Using roulette wheel selection, we compute  $\sum_{j=1}^{\bar{p}} f^j = 10$ , and the probability of selecting  $x^i$  is given by

$$\begin{aligned} \text{prob}(x^1) &= 0.05, & \text{prob}(x^2) &= 0.10, & \text{prob}(x^3) &= 0.40, \\ \text{prob}(x^4) &= 0.15, & \text{prob}(x^5) &= 0.30. \end{aligned}$$

Tournament selection is a complex process, but also stays truest to the biological theme of GA. The idea is to first, from the full population select a (random) sub-population. This subpopulation can be thought of as the population of individuals that have an opportunity to interact. Selection then takes place only within the subpopulation. This makes particular sense when selecting parent individuals, and the subpopulation is often used twice in a row to further fit with the biological theme.

**DEFINITION 4.5 (Tournament Selection).**

Given individuals  $\{x^1, x^2, \dots, x^{\bar{p}}\}$  with corresponding fitness values  $\{f^1, f^2, \dots, f^{\bar{p}}\}$ , and a tournament size  $1 \leq N \leq \bar{p}$ . Tournament selection first randomly chooses  $N$  distinct individuals from  $\{x^1, x^2, \dots, x^{\bar{p}}\}$  with equal probability. The survivor or parent is then selected from this subset of individuals using either elitism or roulette selection.

**EXAMPLE 4.3.** Consider the population of Example 4.2 with tournament size  $N = 2$  and elitism selection. The probability of selecting  $x^1$  is zero, as regardless of which two individuals are chosen,  $x^1$  will never have the higher fitness value:  $\text{prob}(x^1) = 0$ . The probability of selecting  $x^2$  is nonzero, as if the subpopulation is  $\{x^1, x^2\}$ , then  $x^2$  will be selected. Indeed, this is the only way  $x^2$  could be selected, so

$$\text{prob}(x^2) = \frac{1}{C_5^2} = \frac{2! 3!}{5!} = \frac{1}{10}.$$

The probability of selecting  $x^3$  can be computed as

$$\text{prob}(x^3) = \sum_{i=1}^5 \sum_{j=i+1}^5 \text{prob}(x^3|S = \{x^i, x^j\}) \text{prob}(S = \{x^i, x^j\})$$

where  $S$  is used to denote the subpopulation selected. Since  $\text{prob}(S = \{x^i, x^j\}) = 1/C_5^2 = 1/10$  for any  $i, j$ , this simplifies to

$$\begin{aligned} \text{prob}(x^3) &= \frac{1}{10} (\text{prob}(x^3|S = \{x^1, x^2\}) + \text{prob}(x^3|S = \{x^1, x^3\}) + \\ &\quad \text{prob}(x^3|S = \{x^1, x^4\}) + \text{prob}(x^3|S = \{x^1, x^5\}) + \\ &\quad \text{prob}(x^3|S = \{x^2, x^3\}) + \text{prob}(x^3|S = \{x^2, x^4\}) + \\ &\quad \text{prob}(x^3|S = \{x^2, x^5\}) + \text{prob}(x^3|S = \{x^3, x^4\}) + \\ &\quad \text{prob}(x^3|S = \{x^3, x^5\}) + \text{prob}(x^3|S = \{x^4, x^5\})) \\ &= \frac{1}{10} (\text{prob}(x^3|S = \{x^1, x^3\}) + \text{prob}(x^3|S = \{x^2, x^3\}) + \\ &\quad \text{prob}(x^3|S = \{x^3, x^4\}) + \text{prob}(x^3|S = \{x^3, x^5\})) \\ &= \frac{1}{10} (1 + 1 + 1 + 1) = \frac{4}{10}. \end{aligned}$$

Similar computations show that  $\text{prob}(x^4) = \frac{2}{10}$  and  $\text{prob}(x^5) = \frac{3}{10}$ .

---

**EXAMPLE 4.4.** The same example under tournament with roulette wheel selection to select an individual yields

$$\begin{aligned} \text{prob}(x^3) &= \frac{1}{10} (\text{prob}(x^3|S = \{x^1, x^3\}) + \text{prob}(x^3|S = \{x^2, x^3\}) + \\ &\quad \text{prob}(x^3|S = \{x^3, x^4\}) + \text{prob}(x^3|S = \{x^3, x^5\})) \\ &= \frac{1}{10} (4/4.5 + 4/5 + 4/5.5 + 4/7) \approx 0.2987. \end{aligned}$$


---

### 4.3. Encoding

In order to understand Steps 2b) and 2c) of Algorithm 4.1, we must introduce the idea of encoding. In order to stay true to the spirit of genetics, many GA methods begin by “encoding” the points as “chromosomes”. In essence, a point  $x$  is replaced by a binary string  $c \in \{0, 1\}^d$ .

In some applications, this encoding can be quite natural. For example, if one is working with an integer program ( $x \in \mathbb{N}^n$ ) with bound constraints ( $\ell \leq x \leq u$ ), then encoding can be as simple as transforming the integers  $x_i$  into binary strings of fixed length.

**DEFINITION 4.6 (Natural Encoding for Bounded Integer Variables).**

If  $x \in \mathbb{N}^n$  with  $0 \leq \ell_i \leq x_i \leq u_i$  for each  $i = 1, 2, \dots, n$ , then the natural encoding for bounded integer variables replaces each  $x_i$  with the binary string  $c^i \in \{0, 1\}^{d_i}$  that represents  $x_i$ , and where  $d_i$  is the number of bits required to store the upper bound  $u_i$ , as a binary value. The final chromosome is defined by  $c = [c^1, c^2, \dots, c^n]^\top$ .

**EXAMPLE 4.5.** Consider  $S = \{x \in \mathbb{Z}^2 : 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4\}$ . The natural encoding creates two vectors  $c^1 \in \{0, 1\}^2$  and  $c^2 \in \{0, 1\}^3$ . The vector  $x = [1, 4]^\top \in S$  would be encoded as  $[c^1, c^2]^\top = [0, 0, 1, 1, 0]^\top \in \{0, 1\}^5$

Notice that with this encoding, the vector  $c = [0, 0, 1, 1, 0]^\top$  is infeasible because the corresponding value of  $x^2$  would be 6, which exceeds the upper bound of 4. So if the GA ever generates this vector, then that individual must immediately be discarded.

---

There are many other applications where natural encodings occur. These include scheduling problems, path selection problems, and other combinatorial optimization problems. However, this book focuses on problems with continuous variables ( $x \in \mathbb{R}^n$ ), so such natural encodings seldom arise.

One common method of encoding bounded continuous variables is binary discretisation.

**DEFINITION 4.7 (Binary Discretisation Encoding).**

Suppose  $x \in \mathbb{R}^1$  with  $\ell \leq x \leq u$ . Select a positive integer  $d$  and Separate the interval  $[\ell, u]$  into  $2^d$  equally spaced subintervals,

$$[\ell, \ell + \Delta] [\ell + \Delta, \ell + 2\Delta] [\ell + 2\Delta, \ell + 3\Delta] \dots [\ell + (2^d - 1)\Delta, \ell + (2^d)\Delta],$$

where  $\Delta = (u - \ell)/(2^d)$ . Determine which subinterval contains  $x \in [\ell + i\Delta, \ell + (i + 1)\Delta]$ . The chromosome  $c \in \{0, 1\}^d$  is the value  $i$  converted into binary.

If  $x \in \mathbb{R}^n$  with  $\ell \leq x \leq u$ , then each component  $x_i$  can be encoded using binary discretisation to create  $c^i$ , and the final chromosome can be created by stringing all the component chromosomes together  $c = [c^1, c^2, \dots, c^n]^\top$ .

Notice that binary discretisation encoded chromosomes  $c$  only provide information on what subinterval  $x$  lies in, not the exact value of  $x$ . As such, given an encoded chromosome  $c$ , it is not possible to reconstruct a unique  $x$ . To decode  $c$ , we could select a deterministic decoding technique  $x = \ell + (i + \frac{1}{2})\Delta$ , or select a stochastic decoding technique that randomly selects  $x$  in the interval  $[\ell + i\Delta, \ell + (i + 1)\Delta]$ . Binary discretisation encoding loses information.

**EXAMPLE 4.6.** Consider  $\{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 4\}$  and  $d = 3$ . The binary discretisation encoding creates two chromosomes  $c^1 \in \{0, 1\}^3$  and  $c^2 \in \{0, 1\}^3$ . The  $2^3 = 8$  subintervals for  $x_1$  are

$$[0, 3/8) [3/8, 6/8) [6/8, 9/8) \dots [21/8, 24/8].$$

Note that the interval  $[0, 3/8)$  is labelled as subinterval 0 ( $c^1 = [0, 0, 0]^\top$ ), and the interval  $[21/8, 24/8]$  is labelled as subinterval 7 ( $c^1 = [1, 1, 1]^\top$ ). The subintervals for  $x_2$  are

$$[0, 1/2) [1/2, 1) [1, 3/2) \dots [7/2, 4].$$

Hence the point  $[x_1, x_2]^\top = [0.5, 1.1]^\top$  would be encoded as

$$c^1 = [0, 0, 1]^\top, c^2 = [0, 1, 0]^\top, c = [0, 0, 1, 0, 1, 0]^\top.$$

The chromosome  $c = [1, 1, 0, 1, 0, 1]^\top$  could be decoded as any point  $x$  where  $x_1$  belongs to the subinterval 6, and  $x_2$  to subinterval 5:

$$x_1 \in [18/8, 21/8) \quad \text{and} \quad x_2 \in [5/2, 3).$$

Another method for dealing with continuous variables is to leave them unencoded. For the sake of discussion, we shall refer to this as *continuous encoding*. Continuous encoding is clearly the easiest approach, and in practice appears to be more effective. In particular, it does not suffer from the difficulties in decoding binary discretisations. However, it does cause the convergence analysis to be slightly more complex.

#### 4.4. Crossover and Mutation

Given two parent individuals, crossover and mutation aim at creating an offspring individual that acquires attributes from the parents. As is becoming familiar in GA, there are many different crossover and mutation techniques that can be applied. We consider a few of the most popular.

We begin with the crossover techniques that gave “crossover” its name. This technique can be applied to either encoded chromosomes or unencoded points.

**DEFINITION 4.8 (Single/Multi-Point Crossover).**

Suppose parent 1 is represented by  $p = [p_1, p_2, \dots, p_d]^\top$  and parent 2 by  $q = [q_1, q_2, \dots, q_d]^\top$ . To perform a single-point crossover, select a random number  $X \in \{1, 2, \dots, d - 1\}$ , and create the child represented by

$$[p_1, p_2, \dots, p_X, q_{X+1}, q_{X+2}, \dots, q_d]^\top.$$

To perform a 2-point crossover, select 2 random numbers  $X_1 \in \{1, 2, \dots, d - 2\}$  and  $X_2 \in \{X_1 + 1, X_1 + 2, \dots, d - 1\}$  and create the child represented by

$$[p_1, p_2, \dots, p_{X_1}, q_{X_1+1}, q_{X_1+2}, \dots, q_{X_2}, p_{X_2+1}, p_{X_2+2}, \dots, p_d]^\top.$$

To perform a  $n$ -point crossover, select  $n$  random numbers and extend the crossover process in the obvious manner.

**EXAMPLE 4.7.** Suppose  $p = [0, 1, 1, 0, 1, 0, 0, 0]^\top$  and  $q = [1, 1, 0, 0, 0, 1, 0, 1]^\top$ . If a single point crossover is applied, then the potential offspring are

$$(X \in \{1, 2\}) [0, 1, 0, 0, 0, 1, 0, 1]^\top, \quad (X \in \{3, 4\}) [0, 1, 1, 0, 0, 1, 0, 1]^\top \\ (X = 5) [0, 1, 1, 0, 1, 1, 0, 1]^\top, \quad (X \in \{6, 7\}) [0, 1, 1, 0, 1, 0, 0, 1]^\top.$$

Notice,  $X = 1$  and  $X = 2$  give the same offspring (as do  $X \in \{3, 4\}$  and  $X \in \{6, 7\}$ ). Thus, the probability of producing offspring  $[0, 1, 0, 0, 0, 1, 0, 1]^\top$  is  $2/7$ , while the probability of producing offspring  $[0, 1, 1, 0, 1, 1, 0, 1]^\top$  is only  $1/7$ .

Another method to generate offspring is probabilistic gene selection. This technique can also be applied to either encoded chromosomes or unencoded points, but makes more sense when the points are encoded.

**DEFINITION 4.9 (Probabilistic Gene Selection).**

Suppose parent 1 is represented by  $p = [p_1, p_2, \dots, p_d]^\top$  and parent 2 by  $q = [q_1, q_2, \dots, q_d]^\top$ . Select a probability parameter  $\theta \in (0, 1)$ . The probabilistic gene selection technique produces offspring  $c$  such that

$$\text{prob}(c_i = p_i) = \theta \quad \text{and} \quad \text{prob}(c_i = q_i) = (1 - \theta).$$

**EXAMPLE 4.8.** Suppose  $p = [0, 1, 1, 0, 1, 0, 0, 0]^\top$  and  $q = [1, 1, 0, 0, 0, 1, 0, 1]^\top$ . Suppose  $\theta = 0.4$  and probabilistic gene selection is used. Since  $p$  and  $q$  differ in 5 locations, there are  $2^5 = 32$  potential offspring, represented by  $[c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]^\top$  with

$$\begin{aligned} c_1 &\in \{0, 1\}, \text{prob}(c_1 = 0) = \theta; & c_2 &= 1; \\ c_3 &\in \{0, 1\}, \text{prob}(c_3 = 0) = (1 - \theta); & c_4 &= 0; \\ c_5 &\in \{0, 1\}, \text{prob}(c_5 = 0) = (1 - \theta); & c_6 &\in \{0, 1\}, \text{prob}(c_6 = 0) = \theta; \\ c_7 &= 0; \text{ and} & c_8 &\in \{0, 1\}, \text{prob}(c_8 = 0) = \theta. \end{aligned}$$

One example is  $[0, 1, 0, 0, 0, 1, 0, 1]^\top$ , which is produced with probability  $\text{prob}(c = [0, 1, 0, 0, 0, 1, 0, 1]^\top) = \theta \cdot 1 \cdot (1 - \theta) \cdot 1 \cdot (1 - \theta) \cdot (1 - \theta) \cdot 1 \cdot (1 - \theta) = 0.05184$ .

The most probable offspring is  $c = q$ , but even so the probability is only

$$\text{prob}(c = q) = (1 - \theta)^5 = 0.07776.$$

The final method that we will cover is a weighted average technique. This approach only makes sense if the points are unencoded, i.e., the variables remain continuous.

**DEFINITION 4.10 (Weighted Average).**

Suppose parent 1 is represented by  $p = [p_1, p_2, \dots, p_d]^\top$  and parent 2 by  $q = [q_1, q_2, \dots, q_d]^\top$ . Select a weight parameter  $\theta \in (0, 1)$ . The weighted average technique produces the offspring  $c = \theta p + (1 - \theta)q$ .

**EXAMPLE 4.9.** Suppose  $p = [0, 1, 1, 0, 1, 0, 0, 0]^\top$  and  $q = [1, 1, 0, 0, 0, 1, 0, 1]^\top$ . Suppose  $\theta = 0.4$  and weighted average is used. Then the offspring produced will be

$$c = [0.6, 1, 0.4, 0, 0.4, 0.6, 0, 0.6]^\top.$$

Notice, this offspring cannot exist unless the variables remain continuous.

Both probabilistic gene selection and the weighted average require a parameter  $\theta \in (0, 1)$ . The selection of this parameter will obviously affect the outcome of offspring production. If  $\theta \approx 1$ , then the offspring will tend to look more like the first parent. If  $\theta \approx 0$ , then the offspring will tend to look more like the second parent. A value of  $\theta = 1/2$ , provides equal input from each parent. This  $\theta$  parameter could be fixed, randomly selected, or created through some formula. One popular (and easy) method is to fix  $\theta = 1/2$ . Another popular method is to use the fitness values. If the first parent,  $p$ , has fitness  $f^p$  and the second parent,  $q$ , has fitness value  $f^q$ , then  $\theta$  can be set to  $f^p/(f^p + f^q)$ . If the parents have similar fitness, then this will be near 1/2. However, if one of the parents has significantly greater fitness, then  $\theta$  will be near 0 or 1, resulting in offspring that tend to look more like the fitter parent.

In our effort to stay true to the biological theme of GA, after an offspring is created, we proceed to a mutation stage. At this stage, the offspring undergoes some probability of changing slightly. It is important to note that only offspring undergo mutation stage, any survivors are immediately placed in the next population without mutation. The mutation stage allows offspring a better ability to search across the entire constraint set.

As has become a theme in this chapter, there are many options on exactly how to mutate the offspring, and we present only a few of the most popular. The first (bit inversion) is only applicable if individuals are encoded using some form of binary encoding.

**DEFINITION 4.11 (Bit Inversion).**

Suppose crossover has generated offspring  $c$  that is represented by  $c = [c_1, c_2, \dots, c_d]^\top$  where  $c_i \in \{0, 1\}$ . Fix a mutation probability  $\delta \in (0, 1)$ . Then bit inversion mutation creates the mutated offspring  $m$  such that

$$\text{prob}(m_i = c_i) = (1 - \delta) \quad \text{and} \quad \text{prob}(m_i \neq c_i) = \delta.$$

Note the mutation probability  $\delta$  in bit inversion is slight different than the mutation parameter in the GA algorithm. The mutation parameter in the GA algorithm is the probability of a mutation occurring, and can be extracted from the mutation probability in the bit inversion mutation.

**EXAMPLE 4.10.** Suppose  $c = [0, 1, 1, 0]^\top \in \{0, 1\}^4$ . Suppose bit inversion mutation is applied with  $\delta = 1/20$  to create the mutated offspring  $m$ . Then

the probability that  $m_1 = 0$  is  $19/20$  and the probability that  $m_1 = 1$  is  $1/20$ . The probability that  $m = c$  (i.e., no mutation occurs) is

$$\begin{aligned}\text{prob}(m = c) &= \text{prob}(m_1 = c_1) \cdot \text{prob}(m_2 = c_2) \cdot \text{prob}(m_3 = c_3) \\ &\quad \cdot \text{prob}(m_4 = c_4) \\ &= (19/20)^4 \approx 0.8145.\end{aligned}$$

Thus, the mutation parameter in the GA algorithm is  $\gamma = 0.8145$ . The probability that  $m = \mathbf{1} = [1, 1, 1, 1]^\top$  is

$$\begin{aligned}\text{prob}(m = \mathbf{1}) &= \text{prob}(m_1 \neq c_1) \cdot \text{prob}(m_2 = c_2) \cdot \text{prob}(m_3 = c_3) \\ &\quad \cdot \text{prob}(m_4 \neq c_4) \\ &= (19/20)^2(1/20)^2 \approx 0.0023.\end{aligned}$$

If the points are not binary encoded, then a simple variation of bit mutation can be applied.

**DEFINITION 4.12 (Coordinate Perturbation).**

Suppose crossover has generated offspring  $c$  that is represented by  $c = [c_1, c_2, \dots, c_d]^\top$  where  $c_i \in \mathbb{R}$ . Fix a mutation probability  $\delta \in (0, 1)$  and a mutation probability distribution function  $F$ . Then coordinate perturbation mutation creates the mutated offspring  $m$  such that

$$\text{prob}(m_i = c_i) = (1 - \delta) \quad \text{and} \quad \text{prob}(m_i \neq c_i) = \delta,$$

and, if  $m_i \neq c_i$ , then  $m_i = c_i + \xi$  where  $\xi$  follows the mutation probability distribution function  $F$ .

**EXAMPLE 4.11.** Suppose  $c = [0, 1, 1, 0]^\top \in \mathbb{R}^4$ . Suppose coordinate perturbation mutation is applied with  $\delta = 1/20$  and  $F$  being the uniform distribution on the interval  $[-0.1, 0.1]^\top$  (i.e.,  $F(x) = 5$  if  $x \in [-0.1, 0.1]^\top$  and  $F(x) = 0$  otherwise). Then, the probability that  $m = c$  (i.e., no mutation occurs) is as in the previous example. So again, the mutation parameter in the GA algorithm is  $\gamma = 0.8145$ . The probability that  $m_3 > 1.05$  is

$$\begin{aligned}\text{prob}(m_3 > 1.05) &= \text{prob}(m_3 > 1.05 | m_3 \neq c_3) \text{prob}(m_3 \neq c_3) + \\ &\quad \text{prob}(m_3 > 1.05 | m_3 = c_3) \text{prob}(m_3 = c_3) \\ &= 0.25(0.05) + 0(0.95) = 0.0125.\end{aligned}$$

After mutation occurs, it is important to check that the final mutated offspring is still feasible for the optimization problem. If it is not, then the offspring is discarded and the process begins again at the selection stage.

## 4.5. Convergence and Stopping

The GA algorithm does not include any intrinsic measures of distance to convergence (such as a simplex diameter, a step-length, or an approximate gradient – as will be found in the methods later in this book). As a result, there is no clear stopping criterion. Two common stopping criterion include using a function evaluation budget or setting a generation stall limit.

**DEFINITION 4.13 (Evaluation Budget).**

Select a function evaluation budget  $\text{nf}_{\max} > 0$ . The evaluation budget stopping criterion terminates at Step 3 if the number of function evaluations used by GA is greater than  $\text{nf}_{\max}$ .

**DEFINITION 4.14 (Generation Stalling).**

Select a generation stall bound  $\text{gs}_{\max} \in \{2, 3, \dots\}$ . The generation stalling stopping criterion terminates at Step 3 if the GA has found no improvement in function value over the last  $\text{gs}_{\max}$  iterations.

Notice that, as the evaluation budget stopping criterion stops at Step 3, it is possible for the GA to use more than  $\text{nf}_{\max}$  function evaluations before stopping. Indeed, one would expect the final number of function evaluations used will lie somewhere between  $\text{nf}_{\max}$  and  $\text{nf}_{\max} + \bar{p}$ .

In terms of convergence, we first note that GA is, in essence, a stochastic search method. As such, it is impossible to create classical convergence results (such as proving that a limit point must be a critical point). Instead, we shall strive for a probabilistic convergence statement of the form

$$\text{prob} \left( \lim_{k \rightarrow \infty} f_{\text{best}}^k = f^* \right) = 1,$$

where  $f_{\text{best}}^k$  denotes the best objective function value found by iteration  $k$ , and  $f^*$  is the optimal function value.

To achieve convergence we shall need two assumptions.

**ASSUMPTION 4.1 (Monotonicity).** The monotonicity assumption for GA states the best function value for population  $k+1$  is at least as good as the best function value for population  $k$ :

$$\min\{f(x^i) : x^i \in P^{k+1}\} \leq \min\{f(x^i) : x^i \in P^k\}.$$

The monotonicity assumption is easily achieved. Indeed, as long as one survivor is selected using elitist selection, then the monotonicity assumption holds.

We need one of two versions for our second assumption. Both say that the probability of finding a random point is nonzero; however, different phrasing is required for binary encoded and continuous encoded points.

**ASSUMPTION 4.2 (Positive Probabilities for Binary Encoding).** Suppose the points in GA are encoded using a binary encoding. The positive probabilities assumption states that the set of encoded points is finite and there exists  $\varepsilon > 0$  such that

$$\text{prob}(c \in P^k) \geq \varepsilon \quad \text{for any encoded point } c \text{ and iteration } k.$$

**ASSUMPTION 4.3** (Positive Probabilities for Continuous Encoding). *Suppose the points in the GA are encoded using a continuous encoding. The positive probabilities assumption states that the set of encoded points is compact and for every  $\delta > 0$  there exists  $\varepsilon > 0$  such that*

$$\text{prob}(\text{dist}(c, P^k) < \delta) \geq \varepsilon \quad \text{for any point } c \text{ and iteration } k.$$

The positive probabilities assumption is more difficult to ensure than the monotonicity assumption. Nonetheless, it can be achieved in certain circumstances. One example is if bit inversion mutation is applied.

**THEOREM 4.4** (Bit Inversion Implies the Positive Probabilities Assumption).

*Suppose a GA with binary encoding is used. Suppose at least one offspring is generated per iteration. Suppose  $\gamma > 0$  and bit inversion mutation is used. Then the positive probabilities assumption holds.*

**PROOF.** Let  $\bar{c} \in \{0, 1\}^d$  be any binary encoded point. Notice that the number of such points is  $2^d$ , so the set of encoded points is finite. At iteration  $k - 1$ , let  $c$  be the unmuted child of parents  $p$  and  $q$ . Let  $m$  be the mutated child resulting from the bit inversion mutation applied to  $c$ . We consider  $\text{prob}(m = \bar{c}|c)$ .

$$\begin{aligned} \text{prob}(m_i = \bar{c}_i|c) &= \begin{cases} (1 - \gamma) & \text{if } c_i = \bar{c}_i \\ \gamma & \text{if } c_i \neq \bar{c}_i \end{cases} \\ &\geq \min\{\gamma, (1 - \gamma)\}. \end{aligned}$$

Therefore,

$$\text{prob}(m = \bar{c}|c) = \prod_{i=1}^d \text{prob}(m_i = \bar{c}_i|c) \geq (\min\{\gamma, (1 - \gamma)\})^d.$$

So, if we set  $\varepsilon = (\min\{\gamma, (1 - \gamma)\})^d$ , we have

$$\text{prob}(\bar{c} \in P^k) \geq \text{prob}(m = \bar{c}|c) \geq \varepsilon.$$

□

If the GA uses continuously encoded points, the search space is compact, and the mutation probability distribution function is chosen carefully, then it is also possible to prove the positive probabilities assumption holds (see Exercise 4.10).

We now provide a convergence theorem for GA. For the sake of simplicity, we only provide the proof for the binary encoded case. The continuous encoded case is more difficult, but follows a similar flavour to the binary encoded case.

**THEOREM 4.5** (Convergence of GA).

Suppose a GA is used to seek  $\min\{f(x) : x \in \Omega\}$  in such a manner that the monotonicity and positive probability assumptions hold. Let

$$f^* = \min_x \{f(x) : x \in \Omega\}, \quad X^* = \operatorname{argmin}_x \{f(x) : x \in \Omega\}, \\ f_{\text{best}}^k = \min\{f(x^i) : x^i \in P^k\} \text{ and } x_{\text{best}}^k = \operatorname{argmin}\{f(x^i) : x^i \in P^k\}.$$

i. If the points are binary encoded, then

$$\operatorname{prob}\left(\lim_{k \rightarrow \infty} f_{\text{best}}^k = f^*\right) = 1 \quad \text{and} \quad \lim_{k \rightarrow \infty} \operatorname{prob}(P^k \cap X^* \neq \emptyset) = 1.$$

ii. If the points are continuously encoded,  $\Omega$  is compact, and  $f \in \mathcal{C}^0$ , then given a convergent subsequence of  $x_{\text{best}}^{k_i}$  ( $x_{\text{best}}^{k_i} \subset \{x_{\text{best}}^k\}$  with  $x_{\text{best}}^{k_i} \rightarrow \bar{x}$ )

$$\operatorname{prob}\left(\operatorname{dist}(x_{\text{best}}^{k_i}, X^*) \rightarrow 0\right) = 1.$$

**PROOF.** We focus on case i., binary encoded points. Select any  $x^* \in X^*$ . By the positive probabilities assumption,

$$\operatorname{prob}(x^* \in P^k) \geq \varepsilon \quad \text{for } k = 1, 2, \dots$$

This implies, regardless of  $P^{k-1}$

$$\operatorname{prob}(x^* \in P^k | P^{k-1}) \geq \varepsilon \quad \text{for } k = 2, 3, \dots$$

Alternately, for  $k = 2, 3, \dots$ ,  $\operatorname{prob}(x^* \notin P^k | P^{k-1}) < 1 - \varepsilon$ . Hence, for  $k = 2, 3, \dots$ ,  $\operatorname{prob}(X^* \cap P^k = \emptyset | P^{k-1}) < 1 - \varepsilon$ .

Notice, if  $X^* \cap P^{k-1} \neq \emptyset$ , then monotonicity will imply that  $X^* \cap P^k \neq \emptyset$ . In particular,  $\operatorname{prob}(X^* \cap P^k = \emptyset | X^* \cap P^{k-1} \neq \emptyset) = 0$ . Hence,

$$\begin{aligned} \operatorname{prob}(X^* \cap P^k = \emptyset) &= \operatorname{prob}(X^* \cap P^k = \emptyset | X^* \cap P^{k-1} \neq \emptyset) \\ &\quad + \operatorname{prob}(X^* \cap P^{k-1} \neq \emptyset) \\ &= \operatorname{prob}(X^* \cap P^k = \emptyset | X^* \cap P^{k-1} = \emptyset) \\ &\quad + \operatorname{prob}(X^* \cap P^{k-1} = \emptyset) \\ &< 0 + (1 - \varepsilon) \operatorname{prob}(X^* \cap P^{k-1} = \emptyset) \\ &\leq (1 - \varepsilon) \operatorname{prob}(X^* \cap P^{k-1} = \emptyset). \end{aligned}$$

Telescoping, we have  $\operatorname{prob}(X^* \cap P^k = \emptyset) \leq (1 - \varepsilon)^k \operatorname{prob}(X^* \cap P^0 = \emptyset)$ , which proves

$$\lim_{k \rightarrow \infty} \operatorname{prob}(P^k \cap X^* \neq \emptyset) = 1.$$

The other statement in part a) follows immediately.

The statement in part ii. is more technical, but follow analogous ideas.  $\square$

The proof of Theorem 4.5 relies heavily on the positive probabilities assumption, which leads to a reasoning similar to the one used in exhaustive search from Section 3.2. Examining the proof reveals that convergence essentially boils-down to “if you try enough points, eventually you should get

*lucky*”. More recent research into GA methods have developed theorems with weaker assumptions, but similar results. While this may not be a particularly satisfying convergence theorem, it should be noted that GA methods are nonetheless extremely popular and relatively effective in practice.

In practice, encoding techniques, crossover techniques, mutation rates, and population size are key to a GA method’s success. Enormous amounts of research goes into encoding and crossover techniques. In general, the theme seems to be, if you can create an encoding and crossover technique that is very well suited to your problem, then GA can perform well. But, if encoding and crossover are forced on the problem, then it is often better to use one of the other methods in the book. As for mutation, low mutation probabilities are often better, but note that a mutation probability of  $\gamma = 0$  will nullify what little convergence analysis we have. Population size can be very tricky to select, a popular choice is  $\bar{p} = \sqrt{n}$ .

## EXERCISES

---

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 4.1.** Prove that rank fitness and function valued fitness both follow the two rules of fitness.

**EXERCISE 4.2.** Suppose  $x \in \mathbb{R}^3$ ,  $\ell \leq x \leq u$ , where

$$\ell = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \quad \text{and} \quad u = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}.$$

- a) Using binary encoding with  $d = 1$  will separate  $x$  into 8 possible points. List the 8 points and their encodings.
- b) Using binary discretisation encoding with  $d = 3$ , decode the point

$$c = [0, 0, 0, 0, 0, 1, 1, 1]^\top.$$

**EXERCISE 4.3.** Suppose  $x \in [0, 1]$ . Separate the interval  $[0, 1]$  into  $\Upsilon$  equal parts. Use binary encoding to represent the centre point for each part. How long of a binary string is required to encode all options if

- a)  $\Upsilon = 8$ ?
- b)  $\Upsilon = 9$ ?
- c)  $\Upsilon = 16$ ?

**EXERCISE 4.4.** Suppose  $P^k$  has 5 individuals with fitness levels

$$f^1 = 2, \quad f^2 = 2, \quad f^3 = 3, \quad f^4 = 5, \quad f^5 = 8.$$

- a) If tournament selection with roulette wheel is used, what is the probability that chromosome 2 is chosen given  $N = 1$ ,  $N = 2$ , and  $N = 5$ ?
- b) If tournament selection with elitism is used, what is the probability that chromosome 2 is chosen given  $N = 1$ ,  $N = 2$ , and  $N = 5$ ?

**EXERCISE 4.5.** Suppose  $P^k$  has 5 individuals with fitness levels

$$f^1 = 1, \quad f^2 = 2, \quad f^3 = 4, \quad f^4 = 6, \quad f^5 = 6.$$

- a) If tournament selection with roulette wheel is used, what is the probability that chromosome 2 is chosen given  $N = 1$ ,  $N = 2$ , and  $N = 5$ ?
- b) If tournament selection with elitism is used, what is the probability that chromosome 2 is chosen given  $N = 1$ ,  $N = 2$ , and  $N = 5$ ?

**EXERCISE 4.6.** Evaluate

- a)  $\text{prob}(c = [1, 1, 1, 0, 1, 1, 0, 1]^\top)$  in Example 4.8.
- b)  $\text{prob}(m_1 \leq 0)$  in Example 4.11.

**EXERCISE 4.7.** + Suppose  $P^k$  has  $\bar{p}$  individuals with fitness levels and

$$f^1 = 1, f^2 = 2, \dots, f^{\bar{p}} = \bar{p}.$$

Suppose tournament selection with roulette wheel and tournament size  $N$  is used. Prove that the probability of selecting chromosome 1 is strictly decreasing as  $\bar{p}$  increases.

**EXERCISE 4.8.** Suppose  $C$  is encoded using 6 binary bits. Let

$$p^1 = [0, 0, 1, 0, 1, 1]^\top \text{ and } p^2 = [1, 0, 0, 1, 1, 1]^\top,$$

be two parent chromosomes.

- a) Suppose a child is created using single point cross-over, where the cross-over point is randomly selected (with uniform distribution). List all possible child chromosomes, along with the probability of creating each.
- b) Suppose a child is created using probabilistic gene selection, where the averaging probability is  $\theta = \frac{1}{2}$ . List all possible child chromosomes, along with the probability of creating each.

**EXERCISE 4.9.** Suppose  $x$  is bounded in box constraints:  $\ell \leq x \leq u$ , where  $\ell$  and  $u$  are vectors in  $\mathbb{R}^n$ . Another way of encoding  $x \in \mathbb{R}^n$  is the normalised decimal encoding, as follows.

Step 0: Fix an encoding length ( $N$ )

Step 1: Normalise  $x$  to create  $\tilde{x}$  such that  $\mathbf{0} \leq \tilde{x} \leq \mathbf{1}$ .

Step 2: Encode each component of  $\tilde{x}$  using its truncated decimal expansion;

$$\text{e.g., } \tilde{x} = 0.2315634\dots, \quad N = 3 \mapsto c = [2, 3, 1]^\top.$$

Mutation can now be done as follows.

Step 0: Fix a mutation probability  $\delta \in (0, 1)$ .

Step 1: For each component  $c_i$ , mutate with probability  $\delta$ .

Step 2: If mutation occurs, then randomly change  $c_i$  to any other decimal.

Prove that under this encoding, using this discrete mutation technique, Assumption 4.2 (positive probabilities) holds.

**EXERCISE 4.10.** + Suppose  $x$  is bounded in box constraints:  $\ell \leq x \leq u$ , where  $\ell$  and  $u$  are vectors in  $\mathbb{R}^n$ . Suppose the variables use the continuous encoding. Mutation can now be done as follows.

Step 0: Fix a mutation probability  $\delta \in (0, 1)$ .

Step 1: For each individual  $x^i$ , mutate with probability  $\delta$ .

Step 2: If mutation occurs, then randomly change  $x^i$  to any in  $\ell \leq x \leq u$  using a uniform probability distribution function.

Prove that using this mutation technique, Assumption 4.3 (positive probabilities) holds.

**EXERCISE 4.11.** + Suppose  $x$  is bounded in box constraints:  $\ell \leq x \leq u$ , where  $\ell$  and  $u$  are vectors in  $\mathbb{R}^n$ . Suppose the variables use the continuous encoding. Mutation can now be done as follows.

Step 0: Fix a encoding length  $N$  and mutation probability  $\delta \in (0, 1)$ .

Step 1: Apply the encoding of Exercise 4.9 to create  $c$ .

Step 2: Apply the mutation of Exercise 4.9 to  $c$ .

Step 3: If  $c$  is unchanged through mutation, then return  $x$ , otherwise, decode  $c$  and return the decoded variable.

Explain why this mutation does not satisfy Assumption 4.3 (positive probabilities).

**EXERCISE 4.12.** + Prove Theorem 4.5 part ii.




---

**Chapter 4 Project: Apply GA to the Rheology Problem.** Consider the rheology parameter fitting problem in Example 3.1. Experiment different GA using various fitness functions, selection strategies, crossovers, and mutations. Use binary discretisation encodings. Using the tools from the appendix, compare with GS and LHS using a similar number of function evaluations.

# Nelder-Mead

In 1965, John Nelder and Roger Mead published a short (6 pages) note on a new method for unconstrained minimisation. The resulting algorithm has become one of the most widely applied and researched optimization algorithms in the world. Originally, Nelder and Mead titled their method the “Simplex Method”, as the method hinged around using function evaluations at the vertices of a simplex to seek a minimiser. Of course, in the world of optimization, the name “Simplex Method” was already in use – referring to Dantzig’s method algorithm for solving Linear Programs. Thus, in order to avoid confusion, the method became known as the Nelder-Mead (NM) method.

As this chapter is in Part 2 (Heuristics), it should be no surprise that the 1965 variant of NM is an optimization heuristic. However, it should be noted that minor variations to the original Nelder-Mead method creates a new algorithm that conforms to our definition of DFO. That is, convergence to a critical point can be mathematically proven, and mathematical stopping conditions can be established. However, we shall focus on the original Nelder-Mead method.

Another point worth noting is that the Nelder-Mead method is a local search heuristic. That is, the Nelder-Mead method seeks local minimiser to high accuracy. This is the opposite of genetic algorithms, which put greater

focus on seeking a global minimiser, and less focus on high accuracy. As such, when considering global search subroutine to be imbedded into advanced DFO method, Nelder-Mead is not a suitable choice.

### 5.1. The NM Algorithm

Understanding the Nelder-Mead method is highly dependent on simplices, therefore before reading this section, it is critical to have read Section 2.6.

The NM method begins by constructing a starting simplex, and evaluating the objective function at each vertex of the simplex. The function values are then used to rank the vertices, and “reflection”, “expansion”, and “contraction” steps are then used to determine the simplex used in the next iteration. Either the new simplex differs by a single vertex, or has only one vertex in common with the previous simplex.

---

#### ALGORITHM 5.1. Nelder-Mead (NM)

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and the vertices of an initial simplex  $\mathbb{Y}^0 = \{y^0, y^1, \dots, y^n\}$

0. Initialise:

|                            |                               |
|----------------------------|-------------------------------|
| $\delta^e \in (1, \infty)$ | expansion parameter           |
| $\delta^{oc} \in (0, 1)$   | outside contraction parameter |
| $\delta^{ic} \in (-1, 0)$  | inside contraction parameter  |
| $\gamma \in (0, 1)$        | shrink parameter              |
| $k \leftarrow 0$           | iteration counter             |

1. Order:

reorder  $\mathbb{Y}^k$  so  $f(y^0) \leq f(y^1) \leq \dots \leq f(y^n)$  and set  $f_{\text{best}}^k = f(y^0)$

2. Reflect:

set  $x^c = \frac{1}{n} \sum_{i=0}^{n-1} y^i$ , the centroid of all except the worst point  
 set  $x^r = x^c + (x^c - y^n)$  and  $f^r = f(x^r)$   
 if  $f_{\text{best}}^k \leq f^r < f(y^{n-1})$ , then  
     set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^r\}$   
     increment  $k \leftarrow k + 1$  and go to 1

3. Expand:

if  $f^r < f_{\text{best}}^k$ , then test expansion point  
     set  $x^e = x^c + \delta^e(x^c - y^n)$ ,  $f^e = f(x^e)$   
     if  $f^e < f^r$ , then set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^e\}$   
     else ( $f^r \leq f^e$ ), then set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^r\}$   
     increment  $k \leftarrow k + 1$  and go to 1

4a). Outside Contraction:

if  $f(y^{n-1}) \leq f^r < f(y^n)$ , then test outside contraction point  
     set  $x^{oc} = x^c + \delta^{oc}(x^c - y^n)$ ,  $f^{oc} = f(x^{oc})$   
     if  $f^{oc} < f^r$ , then set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^{oc}\}$   
     else ( $f^r \leq f^{oc}$ ), then set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^r\}$   
     increment  $k \leftarrow k + 1$  and go to 1

## 4b). Inside Contraction:

if  $f^r \geq f(y^n)$ , then test inside contraction point  
 set  $x^{ic} = x^c + \delta^{ic}(x^c - y^n)$ ,  $f^{ic} = f(x^{ic})$   
 if  $f^{ic} < f(y^n)$ , then  
 set  $\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, x^{ic}\}$   
 increment  $k \leftarrow k + 1$  and go to 1

## 5. Shrink:

set  $\mathbb{Y}^{k+1} = \{y^0, y^0 + \gamma(y^1 - y^0), y^0 + \gamma(y^2 - y^0), \dots, y^0 + \gamma(y^n - y^0)\}$   
 increment  $k \leftarrow k + 1$  and go to 1

---

Figure 5.1 illustrates the possible trial points generated by the NM algorithm at iteration  $k$ . They are all located on the line passing through the worst point and the centroid  $x^c$  of the opposite face. A compact way of summarising the algorithm consists in comparing  $f^r$  (the objective function value at the reflected point) to the best, second worst, and worst objective function values of the current simplex:  $f_{\text{best}}^k \leq f(y^{n-1}) \leq f(y^n)$ .

- i. If  $f^r < f_{\text{best}}^k$ , then the worst point is replaced by the best of the reflected point  $x^r$  and the expanded point  $x^e$ .
- ii. If  $f_{\text{best}}^k \leq f^r < f(y^{n-1})$ , then the worst point is replaced by the reflected point  $x^r$ .
- iii. If  $f(y^{n-1}) \leq f^r < f(y^n)$ , then the worst point is replaced by the best of  $x^{oc}$  and  $x^r$ .
- iv. If  $f(y^n) \leq f^r$  and  $f^{ic} < f(y^n)$ , then the worst point is replaced by  $x^{ic}$ .
- v. And otherwise, (i.e.,  $f(y^n) \leq f^r$  and  $f(y^n) \leq f^{ic}$ ), then the entire simplex is shrunk, only the best vertex  $y^0$  is kept.

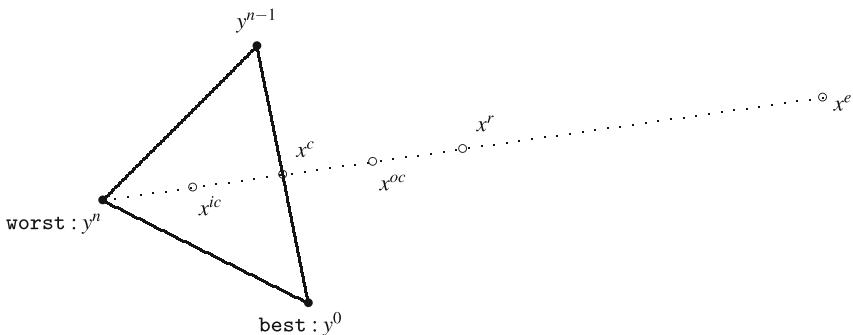


FIGURE 5.1. The possible trial points generated by the NM algorithm, with  $\delta_{ic} = -\frac{1}{2}$ ,  $\delta^{oc} = \frac{1}{2}$  and  $\delta^e = 2$

Examining NM, we see that it is fairly rigid in its description. Except for the ability to select the expansion, outside contradiction, inside contraction, and shrink parameters, very little flexibility is present. This is a stark contrast to GA, where most of the GA algorithm was open to interpretation.

One place where flexibility exists is how ties are broken when ranking points (in Step 1). In most real-world problems, ties are extremely unlikely, so this has little to no effect on algorithm performance.

Despite the apparent lack of flexibility, many minor variations of NM exist. For example, in Step 4a), we present a method that always accepts a new point into the simplex ( $x^r$  or  $x^{oc}$ ). Some variations state that if the outside contraction fails ( $f^r \leq f^{oc}$ ), then the NM algorithm moves to a shrink step. Another variation allows for double expansion step, if the first expansion was particularly successful.

**EXAMPLE 5.1.** Figure 5.2 shows consecutive simplices in  $\mathbb{R}^2$  generated by the NM algorithm. The curves on the figure are contour plots of the objective function. The simplex  $\mathbb{Y}^1$  is obtained through the expansion in Step 3,  $\mathbb{Y}^2$  and  $\mathbb{Y}^4$  through the reflection in Step 2 and  $\mathbb{Y}^3$  through the reflection in Step 3. The next simplex would be  $\mathbb{Y}^5$  obtained through the inside contraction in Step 4b.

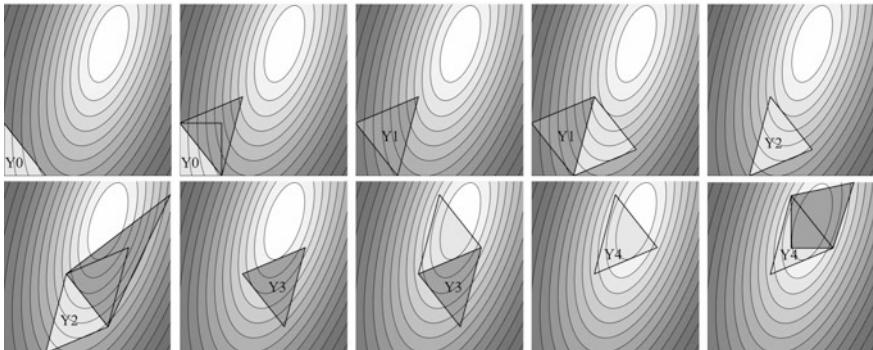


FIGURE 5.2. Simplices in  $\mathbb{R}^2$  generated by the NM algorithm with standard parameters

## 5.2. The Nelder-Mead Simplex

The analysis of the NM algorithm relies on the following theorem, which guarantees that each iteration will be initiated with a simplex.

**THEOREM 5.1 (NM Generates Simplices).**

Suppose NM is applied to minimise  $f$ , and at iteration  $k$  the set  $\mathbb{Y}^k = \{y^0, y^1, \dots, y^n\}$  forms a simplex. Then,  $\mathbb{Y}^{k+1}$  will also form a simplex.

**PROOF.** The set  $\mathbb{Y}^{k+1}$  is produced in one of two manners. Either a shrink step occurred or a nonshrink step occurred. If a shrink step occurs, then

$$\mathbb{Y}^{k+1} = \{y^0, y^0 + \gamma(y^1 - y^0), y^0 + \gamma(y^2 - y^0), \dots, y^0 + \gamma(y^n - y^0)\}.$$

For  $i = 0, 1, 2, \dots, n$ , denote  $y^{i+} = y^0 + \gamma(y^i - y^0)$ . We examine

$$\begin{aligned} L &= [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0] \\ \text{and } L^+ &= [y^{1+} - y^{0+} \quad y^{2+} - y^{0+} \quad \dots \quad y^{n+} - y^{0+}]. \end{aligned}$$

Recall,  $\mathbb{Y}^k$  forms a simplex if and only if  $L$  is invertible (see Theorem 2.5). Thus, as  $\mathbb{Y}^k$  forms a simplex, we know  $\det(L) \neq 0$ . In order to show  $\mathbb{Y}^{k+1}$  forms a simplex, we need to show  $L^+$  is invertible. Examine

$$\begin{aligned} \det(L^+) &= \det[y^{1+} - y^{0+} \quad \dots \quad y^{n+} - y^{0+}] \\ &= \det([y^0 + \gamma(y^1 - y^0) - y^0 \quad \dots \quad y^0 + \gamma(y^n - y^0) - y^0]) \\ &= \det(\gamma[y^1 - y^0 \quad \dots \quad y^n - y^0]) \\ &= \det(\gamma L) \\ &= \gamma^n \det(L) \neq 0. \end{aligned}$$

Hence, if a shrink step occurs, then  $\mathbb{Y}^{k+1}$  forms a simplex.

Alternatively, if a nonshrink step occurs, then  $\mathbb{Y}^{k+1}$  takes the form

$$\mathbb{Y}^{k+1} = \{y^0, y^1, \dots, y^{n-1}, \bar{x}\},$$

where  $\bar{x} = x^c + \delta(x^c - y^n)$  for some  $\delta \in \{\delta^r, \delta^e, \delta^{ic}, \delta^{oc}\}$ . In this case we set  $y^{i+} = y^i$  for  $i = 0, 1, 2, \dots, n-1$  and  $y^{n+} = x^c + \delta(x^c - y^n)$ , and create

$$\begin{aligned} L^+ &= [y^{1+} - y^{0+} \quad y^{2+} - y^{0+} \quad \dots \quad y^{n+} - y^{0+}] \\ &= [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^{n+} - y^0]. \end{aligned}$$

Again, we seek to show  $\det(L^+) \neq 0$ . Using basic determinant relation rules, it can be shown that

$$\det(L^+) = -\delta \det(L) \neq 0,$$

(see Exercise 5.1). □

The formulae derived in Theorem 5.1 also provide us with information on the volume of the simplices generated within the NM method.

### COROLLARY 5.2 (Volumes of the NM Simplices).

Let  $\mathbb{Y}^k$  form a simplex, and suppose NM is used to create  $\mathbb{Y}^{k+1}$ .

i. If a shrink step occurred, then

$$\text{vol}(\mathbb{Y}^{k+1}) = \gamma^n \text{vol}(\mathbb{Y}^k) \quad \text{and} \quad \text{von}(\mathbb{Y}^{k+1}) = \text{von}(\mathbb{Y}^k).$$

ii. If an nonshrink step occurred, then

$$\text{vol}(\mathbb{Y}^{k+1}) = |\delta| \text{vol}(\mathbb{Y}^k),$$

where  $\delta \in \{\delta^r, \delta^e, \delta^{ic}, \delta^{oc}\}$  as appropriate.

**PROOF.** We use the notation from the proof of Theorem 5.1.

i. If a shrink step occurred, then

$$\text{vol}(\mathbb{Y}^{k+1}) = \left| \frac{\det(L^+)}{n!} \right| = \left| \frac{\gamma^n \det(L)}{n!} \right| = \gamma^n \text{vol}(\mathbb{Y}^k).$$

The diameter  $\text{diam}(\mathbb{Y})$  is required to compute the normalised volume  $\text{von}(\mathbb{Y}^{k+1})$ :

$$\begin{aligned}\text{diam}(\mathbb{Y}^{k+1}) &= \max_{i \neq j} \|y^{i+} - y^{j+}\| = \max_{i \neq j} \|y^0 + \gamma(y^i - y^0) - (y^0 + \gamma(y^j - y^0))\| \\ &= \gamma \max_{i \neq j} \|y^i - y^j\| = \gamma \text{diam}(\mathbb{Y}^k).\end{aligned}$$

The normalised volume is

$$\text{von}(\mathbb{Y}^{k+1}) = \frac{\text{vol}(\mathbb{Y}^{k+1})}{(\text{diam}(\mathbb{Y}^{k+1}))^n} = \frac{\gamma^n \text{vol}(\mathbb{Y}^k)}{(\gamma \text{diam}(\mathbb{Y}^k))^n} = \frac{\text{vol}(\mathbb{Y}^k)}{\text{diam}(\mathbb{Y}^k)^n} = \text{von}(\mathbb{Y}^k).$$

ii. If a nonshrink step occurred, then

$$\text{vol}(\mathbb{Y}^{k+1}) = \left| \frac{\det(L^+)}{n!} \right| = \left| \frac{-\delta \det(L)}{n!} \right| = |\delta| \text{vol}(\mathbb{Y}^k),$$

where  $\delta \in \{\delta^r, \delta^e, \delta^{ic}, \delta^{oc}\}$  as appropriate.  $\square$

It is important to note that Corollary 5.2 does not provide the normalised volume of the new simplex if a nonshrink step occurred. Indeed, there is no general formula for the normalised volume after a nonshrink step. Even in the specific case of a reflection step, the normalised volume can increase, decrease, or stay the same (see Exercise 5.6).

### 5.3. Convergence Study

As a heuristic method, very little can be said about the convergence of the NM method. Indeed, when reading the next theorem, note that it essentially states “NM converges, but not necessarily to a critical point”.

**THEOREM 5.3 (Convergence of NM).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be bounded below. Suppose a NM is used to seek  $\min\{f(x) : x \in \mathbb{R}^n\}$ . At iteration  $k$ , denote  $\mathbb{Y}^k = \{y^{0,k}, y^{1,k}, \dots, y^{n,k}\}$  ordered so that  $f(y^{0,k}) \leq f(y^{1,k}) \leq \dots \leq f(y^{n,k})$ . Let  $x_{\text{best}}^k = y^{0,k}$  and  $f_{\text{best}}^k = f(y^{0,k})$ .

- i.  $f_{\text{best}}^k$  converges to some value  $\hat{f}$ .
- ii. If a finite number of nonshrink steps occur, then  $x_{\text{best}}^k$  converges to some point  $\hat{x}$ , and all vertices  $y^{i,k} \in \mathbb{Y}^k$  also converge to  $\hat{x}$ .
- iii. If a finite number of shrink steps occur, then for each fixed  $i = 0, 1, \dots, n$  the values  $f(y^{i,k})$  converge to some value  $\hat{f}^i$ . Moreover, these values satisfy  $\hat{f}^0 \leq \hat{f}^1 \leq \dots \leq \hat{f}^n$ .

**PROOF.** i. Note that  $f_{\text{best}}^k$  is nonincreasing (by algorithm design) and bounded below (as  $f$  is bounded below). As such  $f_{\text{best}}^k$  converges to some value  $\hat{f}$ .

ii. Suppose the final nonshrink steps occurs at iteration  $\bar{k}$ . So, for all  $k > \bar{k}$ , we have

$$y^{0,k+1} = y^{0,k} \quad y^{1,k+1} = y^{0,k} + \gamma(y^{1,k} - y^{0,k}) \quad \dots \quad y_{k+1}^n = y^{0,k} + \gamma(y^{n,k} - y^{0,k}).$$

Clearly,  $y^{0,k}$  converges to  $\hat{x} = y^{0,\bar{k}}$ . Now, to examine  $y^{i,k}$ , we first note that whenever  $k > \bar{k}$ , we have

$$\begin{aligned} \|y^{i,k+1} - y^{0,\bar{k}}\| &= \|y^{0,k} + \gamma(y^{i,k} - y^{0,k}) - y^{0,\bar{k}}\| \\ &= \|y^{0,\bar{k}} + \gamma(y^{i,k} - y^{0,\bar{k}}) - y^{0,\bar{k}}\| \\ &= \gamma\|y^{i,k} - y^{0,\bar{k}}\| \end{aligned}$$

Telescoping this event, we note

$$\|y^{i,\bar{k}+m} - y^{0,\bar{k}}\| = \gamma^m \|y^{i,\bar{k}} - y^{0,\bar{k}}\|.$$

As  $\gamma \in (0, 1)$ , this implies  $\lim_{k \rightarrow \infty} y^{i,k} = y^{0,\bar{k}} = \hat{x}$ .

iii. If  $k$  is a nonshrink step, then  $f(y^{i,k+1}) \leq f(y^{i,k})$  for all  $i$ . Thus, after all shrink steps are complete, then each sequence  $f(y^{i,k})$  is nonincreasing and bounded below. Thus all of the sequences converge. The ordering of  $\hat{f}^i$  follows from the fact that  $f(y^{0,k}) \leq f(y^{1,k}) \leq \dots \leq f(y^{n,k})$  for all  $k$ .  $\square$

#### 5.4. The McKinnon Example

As mentioned, Theorem 5.3 does not provide much insight on when NM converges to a solution to the optimization problem. In practice, NM is fairly effective, as this resulted in many conjectures on it converging to a solution when the problem is nicely behaved. However, in 1998 (33 years after the publication of the algorithm), McKinnon published an example of a convex  $\mathcal{C}^1$  function for which the NM algorithm converges to a non-minimiser. Reconstructing this example provides an excellent insight on the process of mathematical research.

Our first step in reconstructing McKinnon's example is to decide how to make NM fail. After some experimentation, it becomes reasonably clear that the best way to make NM fail is to make the normalised volume of the simplex  $\mathbb{Y}^k$  tend to zero. When this happens, NM ceases to obtain information in all dimensions of the space, and can thereby miss good descent directions.

McKinnon's idea was to create an example where NM would follow an infinite sequence of inside contraction steps. This will make the normalised volume tend to zero, and have the bonus of keeping  $x_{\text{best}}^k = y^0$  constant throughout the implementation.

Note that it is easy to prove NM converges when applied to a smooth convex function in  $\mathbb{R}^1$  (Exercise 5.4). Therefore, to make a concrete example, let us fix the dimension as  $\mathbb{R}^2$ , and use the standard default parameters of NM:  $\gamma = \frac{1}{2}$ ,  $\delta^e = 2$ ,  $\delta^{ic} = -\frac{1}{2}$ ,  $\delta^{oc} = \frac{1}{2}$ . If we want iteration  $k$  to produce an inside contraction step, then we require the following situation to occur

$$\begin{array}{lll} f(x^r) &\geq& f(y^n) &\text{i.e., the reflection point is rejected,} \\ f(x^{ic}) &<& f(y^n) &\text{i.e., the inside contraction point is accepted.} \end{array}$$

In addition, in order to make each simplex  $\mathbb{Y}^k$  have the longest side correspond to  $y^0y^n$ , we need to force a reordering to occur at every iteration. So we require

$$f(x^{ic}) < f(y^1) \quad \text{i.e., the inside contraction point is the next } y^1.$$

Figure 5.3 illustrates the points evaluated during an inside contraction step.

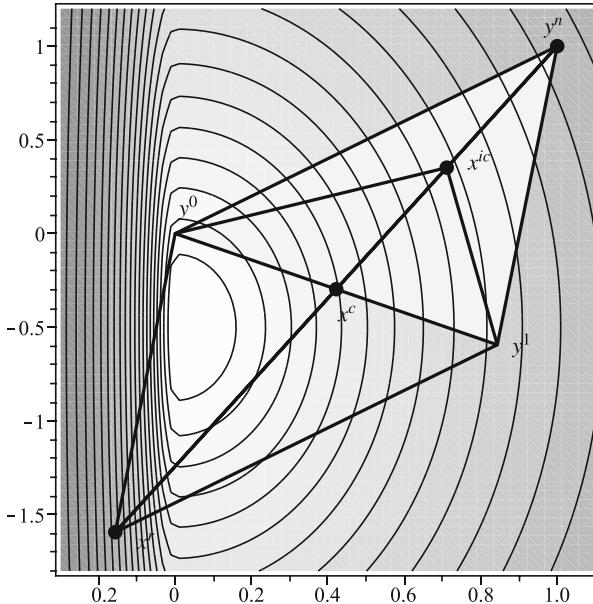


FIGURE 5.3. The points  $x^r$  and  $x^{ic}$  are evaluated by NM when an inside contraction takes place. The simplex goes from  $\{y^0, y^1, y^n\}$  to  $\{y^0, x^{ic}, y^n\}$

To make things more concrete, let us set  $y^0 = [0, 0]^\top$ . At iteration  $k$ , denote  $\mathbb{Y}^k = \{y^{0,k}, y^{1,k}, y^{2,k}\}$ . We desire

$$\begin{aligned} y^{0,k+1} &= y^{0,k} = [0, 0]^\top, \\ y^{1,k+1} &= x^{ic}, \\ y^{2,k+1} &= y^{1,k}. \end{aligned}$$

Notice that the centroid of the face opposite to the worst point is  $x^c = \frac{1}{2}(y^{1,k} - y^{0,k}) = \frac{1}{2}y^{1,k}$ , so

$$\begin{aligned} y^{1,k+1} = x^{ic} &= x^c + \delta^{ic}(x^c - y^{2,k}) \\ &= \frac{1}{2}y^{1,k} - \frac{1}{2}(\frac{1}{2}y^{1,k} - y^{2,k}) \\ &= \frac{1}{4}y^{1,k} + \frac{1}{2}y^{2,k} \\ &= \frac{1}{4}y^{1,k} + \frac{1}{2}y^{1,k-1}. \end{aligned}$$

So, in order to create our example, we need to satisfy the following *second order recurrence relation*

$$0 = y^{1,k+1} - \frac{1}{4}y^{1,k} - \frac{1}{2}y^{1,k-1}.$$

To approach this, we use the following (classical) lemma regarding second order recurrence relations.

**LEMMA 5.4 (Solutions to Recurrence Relations).**

Let  $A$  and  $B$  be real numbers. Consider the second order recurrence given by

$$(5.1) \quad 0 = x_{k+1} - Ax_k - Bx_{k-1}.$$

The sequence  $\{1, t, t^2, t^3, \dots\}$  satisfies relation (5.1) if and only if  $t$  solves  $t^2 - At - B = 0$ .

**PROOF.** Suppose  $t$  is a solution of the quadratic equation  $t^2 - At - B = 0$ , and consider  $x_k = t^k$ . Examining  $x_{k+1} - Ax_k - Bx_{k-1}$  we see

$$\begin{aligned} x_{k+1} - Ax_k - Bx_{k-1} &= t^{k+1} - At^k - Bt^{k-1} \\ &= t^{k-1}(t^2 - At - B) = 0. \end{aligned}$$

Conversely, suppose the sequence  $\{x_k\}$  takes the form  $\{1, t^1, t^2, \dots\}$  and satisfies  $x_{k+1} - Ax_k - Bx_{k-1} = 0$ , then setting  $k = 1$  shows  $(t^2 - At - B) = 0$ .  $\square$

Returning to  $0 = y^{1,k+1} - \frac{1}{4}y^{1,k} - \frac{1}{2}y^{1,k-1}$ , applying Lemma 5.4 we solve  $t^2 - \frac{1}{4}t - \frac{1}{2} = 0$ , to find

$$t = \frac{1 \pm \sqrt{33}}{8}.$$

As these roots will be used repeatedly, let us define

$$\lambda = \frac{1 + \sqrt{33}}{8} \approx 0.8431 \quad \text{and} \quad \mu = \frac{1 - \sqrt{33}}{8} \approx -0.5931.$$

Now, in order for NM to create an infinite series of inside contractions, we shall use

$$\begin{aligned} \mathbb{Y}^0 &= \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda \\ \mu \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \\ \mathbb{Y}^1 &= \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda^2 \\ \mu^2 \end{bmatrix}, \begin{bmatrix} \lambda \\ \mu \end{bmatrix} \right\} \\ &\vdots \\ \mathbb{Y}^k &= \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda^{k+1} \\ \mu^{k+1} \end{bmatrix}, \begin{bmatrix} \lambda^k \\ \mu^k \end{bmatrix} \right\}. \end{aligned}$$

Using these sets, at iteration  $k$  we will have

$$\begin{aligned} x^c &= \frac{1}{2}(y^{1,k} + y^{0,k}) = \frac{1}{2} \begin{bmatrix} \lambda^{k+1} \\ \mu^{k+1} \end{bmatrix} \\ x^r &= x^c + (x^c - y^{2,k}) = \begin{bmatrix} \lambda^k(\lambda - 1) \\ \mu^k(\mu - 1) \end{bmatrix} \\ x^{ic} &= y^{1,k+1} = \begin{bmatrix} \lambda^{k+2} \\ \mu^{k+2} \end{bmatrix}. \end{aligned}$$

We require that the reflection fails:  $f(x^r) > f(y^{2,k})$  but the inside contraction succeeds:  $f(x^{ic}) < f(y^{1,k})$ .

Notice that the first coordinate of  $x^r$  is  $\lambda^k(\lambda - 1)$ , which is always negative (since,  $0 < \lambda < 1$ ). However, the first coordinate of  $x^{ic}$  is  $\lambda^{k+2}$ , which is always positive. So, to create our function  $f$ , we want  $f(x_1, x_2)$  to be very large when  $x_1 < 0$  and well behaved when  $x_1 > 0$ .

Consider the functional form

$$f(x_1, x_2) = \begin{cases} \theta_{big}(x_1)^2 + g(x_2) & \text{if } x_1 \leq 0 \\ \theta_{small}(x_1)^2 + g(x_2) & \text{if } x_1 > 0, \end{cases}$$

where  $\theta_{big}$  is a large constant,  $\theta_{small}$  is a small positive constant, and  $g$  is a smooth convex function that depends only on  $x_2$ . If the parameters  $\theta_{big}$  and  $\theta_{small}$  are selected carefully, we should be able to satisfy  $f(x^r) > f(y^{2,k})$  and  $f(x^{ic}) < f(y^{1,k})$ . If  $g$  is smooth, convex, fairly flat (so as not to affect  $f(x^r) > f(y^{2,k})$  and  $f(x^{ic}) < f(y^{1,k})$ ), and not minimised at  $[0, 0]^\top$ , then we should have the counter example.

The following Lemma provides useful inequalities to support the counter example. Exercise 5.11 breaks down its proof in five steps.

#### LEMMA 5.5 (Inequalities for McKinnon's Example).

Let  $\mu = \frac{1-\sqrt{33}}{8} < 0 < \lambda = \frac{1+\sqrt{33}}{8}$  and  $k \in \mathbb{N}$ . Then the following inequalities hold

$$(5.2) \quad 6\lambda^{2k+2}(\lambda^2 - 1) < \mu^{k+1}(1 - \mu)(1 + \mu^{k+1}(1 + \mu))$$

$$(5.3) \quad \lambda^{2k}(360(\lambda - 1)^2 - 6) > \mu^k(2 - \mu)(1 + \mu^{k+1}) > 0.$$

The following proposition shows that taking  $\theta_{small} = 6$  and the convex function  $g \in \mathcal{C}^\infty$  as  $g(b) = b + b^2$  leads to a successful inside contraction.

#### PROPOSITION 5.6.

If  $\theta_{small} = 6$  and  $g(b) = b + b^2$  then  $f(y^0) < f(x^{ic}) < f(y^{1,k})$ .

**PROOF.** Setting  $b = \mu^{k+1}$  yields

$$\begin{aligned} f(x^{ic}) - f(y^{1,k}) &= f(\lambda^{k+2}, \mu^{k+2}) - f(\lambda^{k+1}, \mu^{k+1}) \\ &= \theta_{small}\lambda^{2k+4} + g(\mu b) - \theta_{small}\lambda^{2k+2} - g(b) \\ &= 6\lambda^{2k+2}(\lambda^2 - 1) + g(\mu b) - g(b). \end{aligned}$$

Expanding the two last terms gives

$$\begin{aligned} g(\mu b) - g(b) &= (\mu^2 - 1)b^2 + (\mu - 1)b \\ &= (\mu - 1)b((\mu + 1)b + 1) \\ &= \mu^{k+1}(\mu - 1)((\mu + 1)\mu^{k+1} + 1) \\ &< 6\lambda^{2k+2}(1 - \lambda^2) \quad \text{by Equation (5.2).} \end{aligned}$$

Substituting in the previous equality shows that  $f(y^0) = 0 < f(x^{ic}) < f(y^{1,k})$ , the inside contraction is successful.  $\square$

The only remaining parameter to be identified is  $\theta_{big}$ . The following proposition shows that taking it to be equal to 360 will lead to an unsuccessful reflection step, as required.

**PROPOSITION 5.7.**

If  $\theta_{small} = 6$ ,  $g(b) = b + b^2$  and  $\theta_{big} = 360$  then  $f(x^r) > f(y^{2,k})$ .

**PROOF.** Setting  $b = \mu^k$  yields

$$\begin{aligned} f(x^r) - f(y^{2,k}) &= f(\lambda^k(\lambda - 1), \mu^k(\mu - 1)) - f(\lambda^k, \mu^k) \\ &= \theta_{big}\lambda^{2k}(\lambda - 1)^2 + g(b(\mu - 1)) - \theta_{small}\lambda^{2k} - g(b) \\ &= \lambda^{2k}(360(\lambda - 1)^2 - 6) + b^2(\mu^2 - 2\mu + 1) + b(\mu - 1) - b^2 - b \\ &= \lambda^{2k}(360(\lambda - 1)^2 - 6) + b(\mu - 2)(b\mu + 1) \\ &= \lambda^{2k}(360(\lambda - 1)^2 - 6) + \mu^k(\mu - 2)(\mu^{k+1} + 1) \\ &< 0 \quad \text{by Equation (5.3).} \end{aligned}$$

This shows that the reflection step is unsuccessful.  $\square$

In summary, applying NM with the standard parameter to the convex  $\mathcal{C}^1$  function

$$f(x_1, x_2) = \begin{cases} 360(x_1)^2 + x_2 + (x_2)^2 & \text{if } x_1 \leq 0 \\ 6(x_1)^2 + x_2 + (x_2)^2 & \text{if } x_1 > 0. \end{cases}$$

from the initial simplex

$$\mathbb{Y}^0 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda \\ \mu \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

results in an infinite series of inside contractions, so

$$\mathbb{Y}^k = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda^{k+1} \\ \mu^{k+1} \end{bmatrix}, \begin{bmatrix} \lambda^k \\ \mu^k \end{bmatrix} \right\}.$$

This results in all vertices of the simplex converging to  $[0, 0]^\top$ , but the function is minimised at  $[0, -\frac{1}{2}]^\top$ . Figure 5.4 plots the first five iterations of NM applied to this function.

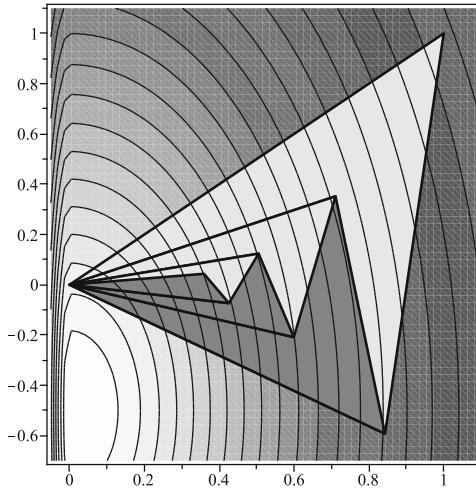


FIGURE 5.4. Five NM iterations on the McKinnon example

From McKinnon’s example, we can deduce the flaw in the NM algorithm. In particular, in Section 5.4, we were able to drive the normalised volume to 0 and thereby create an example where NM converged to a non-minimiser. In Exercises 5.6 and 5.9, it is shown that reflection, expansion, and contraction steps can all reduce the normalised volume of the simplex, so any of these steps can lead to poor behaviour of the NM algorithm. As the normalised volume approaches 0, NM will have a harder and harder time searching all directions of the objective space, and thus becomes more likely to fail.

Several methods have been proposed to correct this weakness. One of them is presented in the Further Remarks on page 90, in which the NM algorithm is adapted so that if the normalised volume of  $\mathbb{Y}^{k+1}$  becomes poor, then the trial point is rejected and an alternate step is taken.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

$$\begin{aligned}\text{EXERCISE 5.1. Prove } \det & \left( \begin{bmatrix} y^1 - y^0 & y^2 - y^0 & \dots & x^c + \delta(x^c - y^n) - y^0 \end{bmatrix} \right) \\ &= -\delta \det \left( \begin{bmatrix} y^1 - y^0 & y^2 - y^0 & \dots & y^n - y^0 \end{bmatrix} \right)\end{aligned}$$

when  $x^c = \frac{1}{n} \sum_{i=0}^{n-1} y^i$ .

[Hint: use the linear algebra property that adding a multiple of one column to a different column does not change the determinant.]

**EXERCISE 5.2.** Provide a breakdown of how many function evaluations the NM algorithm will use during a given iteration. (Your answer will involve if statements.)

**EXERCISE 5.3.** Explain why Nelder and Mead chose not to evaluate  $f$  at the centroid  $x^c$ .

**EXERCISE 5.4.** Let  $f : \mathbb{R} \mapsto \mathbb{R}$  be a convex function of 1 variable. Show that  $f_{\text{best}}^k$  (produced by NM on  $f$ ) either converges to the minimal value of  $f$  or that it goes to  $-\infty$ .

**EXERCISE 5.5.** + Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be a strictly convex function, i.e.,  $f(\theta x + (1 - \theta)y) < \theta f(x) + (1 - \theta)f(y)$  for all  $x, y \in \mathbb{R}^n$  and  $\theta \in (0, 1)$ .

- Prove that applying the NM algorithm on  $f$  will never require a shrink step.
- Suppose that  $f : \mathbb{R}^n \rightarrow \mathbb{R}_+$  is a strictly convex function. Show that the function  $g(x) = \sqrt{f(x)}$  is not necessarily convex.
- Prove that applying the NM algorithm on  $g$  (from question b) will never require a shrink step.

**EXERCISE 5.6.** Reflection steps can change the normalised volume of a simplex. To see this, suppose that the NM algorithm has formed the ordered simplex points  $\mathbb{Y}^k = \{y^0, y^1, y^2, y^3\} \subseteq \mathbb{R}^3$ , where

$$y^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad y^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad y^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad y^3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

- List the possible output simplices generated after one iteration of the NM algorithm.
- Suppose that the NM algorithm creates a new simplex  $\mathbb{Y}^{k+1}$  by a reflection step. Find  $\text{von}(\mathbb{Y}^k)$  and  $\text{von}(\mathbb{Y}^{k+1})$ .

**EXERCISE 5.7.** + Is it true that if the centroid  $x^c$  is the origin, then the normalised volumes of a simplex and its reflection are identical?

**EXERCISE 5.8.** + Suppose that the NM algorithm has formed the ordered simplex points  $\mathbb{Y}^k = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  and produces a new simplex  $\mathbb{Y}^{k+1}$  by a reflection step. Find  $\text{von}(\mathbb{Y}^k)$  and  $\text{von}(\mathbb{Y}^{k+1})$  for the following.

- $y^n = 0$  and for every  $i$  from 0 to  $n - 1$ , set  $y^i = e_{i+1}$ , where  $e_i$  is the  $i$ -th coordinate vector.
- $y^0 = 0$  and for every  $i$  from 1 to  $n$ , set  $y^i = y^{i-1} + e_i$ , where  $e_i$  is the  $i$ -th coordinate vector.

**EXERCISE 5.9.**  $\square$  Nonshrink steps can increase or decrease the normalised volume of a simplex. For  $0 < \theta < \frac{\pi}{2}$ , consider the simplex

$$\mathbb{Y}^0 = \{[0, 0]^\top, [\cos \theta, \sin \theta]^\top, [\cos \theta, -\sin \theta]^\top\}.$$

Let  $\mathbb{Y}^r$ ,  $\mathbb{Y}^e$ ,  $\mathbb{Y}^{ic}$  and  $\mathbb{Y}^{oc}$  be the simplices obtained by replacing the origin by a reflection, an expansion, an inside contraction, and an outside contraction using the standard default parameters of NM:  $\delta^e = 2$ ,  $\delta^{ic} = -\frac{1}{2}$ ,  $\delta^{oc} = \frac{1}{2}$ .

- Show that  $\text{von}(\mathbb{Y}^r) = \text{von}(\mathbb{Y}^0)$ .
- Show that  $\text{von}(\mathbb{Y}^{ic}) = \text{von}(\mathbb{Y}^{oc})$ .

- c) On a single graph, plot the volume (vol) of  $\mathbb{Y}^0, \mathbb{Y}^e$  and  $\mathbb{Y}^{ic}$  as a function of  $\theta$ .
- d) On a single graph, plot the diameter (diam) of  $\mathbb{Y}^0, \mathbb{Y}^e$  and  $\mathbb{Y}^{ic}$  as a function of  $\theta$ .
- e) On a single graph, plot the normalised volume (von) of  $\mathbb{Y}^0, \mathbb{Y}^e$  and  $\mathbb{Y}^{ic}$  as a function of  $\theta$ .

**EXERCISE 5.10.** + Consider the McKinnon Function

$$f(x_1, x_2) = \begin{cases} 360(x_1)^2 + x_2 + (x_2)^2 & \text{if } x_1 \leq 0 \\ 6(x_1)^2 + x_2 + (x_2)^2 & \text{if } x_1 > 0. \end{cases}$$

- a) Prove  $f$  is  $\mathcal{C}^1$ .
- b) Prove  $f$  is convex.
- c) Find the minimiser of  $f$ .

[Hint: first prove that the minimiser occurs when  $x_1 = 0$ , then use calculus to minimise  $x_2 + (x_2)^2$ .]

**EXERCISE 5.11.** + The objective of this exercise is to prove Lemma 5.5 by breaking down the proof in easier steps. Show

- a)  $0 < 1 - \mu < 6(1 - \lambda^2)$ .
  - b)  $8 - \mu < 360(\lambda - 1)^2$ .
  - c)  $\lambda^{2k} > |\mu|^k \geq \mu^k$ .
  - d) Equation (5.2) :  $6\lambda^{2k+2}(\lambda^2 - 1) < \mu^{k+1}(1 - \mu)(1 + \mu^{k+1}(1 + \mu))$ .
  - e) Equation (5.3) :  $\lambda^{2k}(360(\lambda - 1)^2 - 6) > \mu^k(2 - \mu)(1 + \mu^{k+1}) > 0$ .
- [Hint: split into two cases,  $k$  odd or even.]

**EXERCISE 5.12.** + Consider the simplex produced by NM at iteration  $k$  on the McKinnon function

$$\mathbb{Y}^k = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \lambda^{k+1} \\ \mu^{k+1} \end{bmatrix}, \begin{bmatrix} \lambda^k \\ \mu^k \end{bmatrix} \right\}.$$

where  $\lambda = \frac{1}{8}(1 + \sqrt{33})$  and  $\mu = \frac{1}{8}(1 - \sqrt{33})$ .

- a) Evaluate  $\text{vol}(\mathbb{Y}^k)$  and  $\text{vol}(\mathbb{Y}^{k+1})/\text{vol}(\mathbb{Y}^k)$ .
- b) Evaluate  $\text{diam}(\mathbb{Y}^k)$  and  $\lim_{k \rightarrow \infty} \text{diam}(\mathbb{Y}^k)$ .
- c) Evaluate  $\text{von}(\mathbb{Y}^k)$  and  $\lim_{k \rightarrow \infty} \text{von}(\mathbb{Y}^k)$ .

□

**Chapter 5 Project: Apply NM to the Rheology Problem.** Apply the NM algorithm on the nonsmooth rheology problem from Section 1.3.3 using the formulation from Example 3.1.

- a) Use the initial simplex  $\mathbb{Y}^0 = \{0, e_1, e_2, e_3\}$  (the origin together with the three coordinate directions) and the standard parameters  $\delta^e = 2$ ,  $\delta^{ic} = -\frac{1}{2}$ ,  $\delta^{oc} = \frac{1}{2}$  and  $\gamma = \frac{1}{2}$ .

- b) Use the simplex from question a), translated by  $[9.5, 9.5, 9.5]^\top$  using again the standard parameters.
- c) Propose and use another set of NM parameters.
- d) Generate convergence plots (Appendix Section A.3.1) to compare the runs of the three previous questions (use a logarithmic scale to improve the graph readability).

# *Further Remarks on Heuristics*

---

The GA algorithm presented in Chapter 4 is an example of an Evolutionary strategy. Evolutionary strategies date back to at least 1971, when Rechenberg published his Ph.D. thesis “Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution” (in German) [144]. (Although some sources suggest Schwefel should be credited with the idea [28].) Holland’s book, “Adaptation in Natural and Artificial Systems” (1975) [98], is widely regarded as the source for the GA algorithm’s popularity as a modern research tool. Since then, various Evolutionary algorithms have been implemented, adapted, and applied by numerous researchers. It would be impossible to even scratch the surface of the vast research in the field, so even survey papers in the area tend to focus on specific applications such as Evolutionary algorithms for: control systems [72], data mining [78], clustering problems [101], earth imaging [61], and multiobjective optimization [174].

It should also be noted that the GA algorithm is by no means the only Evolutionary strategy, and Evolutionary algorithms are certainly not the only heuristic approach to BBO [89, 95]. For example, Particle swarm optimization [106] and Simulated annealing [107] have both shown success in BBO and grown in popularity in recent years. Comparison of a direct search method and a GA for conformational searching problem is presented in [127, 126].

Modern research often links Evolutionary algorithms (or other heuristic search methods) with some other DFO method to ensure a balance of a global search heuristic with a stronger local convergence analysis. How this is accomplished is discussed in Parts 3 and 4 of this book.

There is no dispute to the origin of the Nelder-Mead algorithm. The brief (6-page) 1965 paper “A Simplex Method for Function Minimization”, by Nelder and Mead, first introduced the method to the world [132]. The name *Simplex method* was quickly dropped, as Dantzig’s work in linear pro-

gramming had already adopted the name [55]. Since then, the method has become highly popular as a quick and effective method for approaching real-world BBO problems [95, 171, 172].

In 1987, Dennis and Wood [60] created a strictly convex function where a minor adaptation of the NM algorithm failed to converge. In 1994, Strasser's Ph.D. thesis [157] provides a nonconvex example where the (unaltered) NM algorithm fails to converge. The example given in Section 5.4 is based on the work of McKinnon [125]. However, McKinnon's work was actually more general. The original example uses

$$f(x_1, x_2) = \begin{cases} \theta_{big}|x_1|^\tau + g(x_2) & x \leq 0 \\ \theta_{small}|x_1|^\tau + g(x_2) & x > 0, \end{cases}$$

and

$$\theta_{small} \left( \frac{\theta_{big}}{\theta_{small}} (1 - \lambda_1)^\tau - 1 \right) > 2 - \lambda_2,$$

where  $\tau$  is any integer greater than or equal to 2. This shows it is possible to make a  $\mathcal{C}^k$  counter example.

Two papers by Wright [171, 172] position Nelder-Mead and Mckinnon's example within a broader context. In the latter, she quotes John Nelder from an interview conducted in 2000:

There are occasions where it [the NM algorithm] has been spectacularly good... Mathematicians hate it because you cannot prove convergence; engineers seem to love it because it often works.

In 1999, Tseng proposed the fortified NM algorithm [162] to guarantee convergence to a critical point, a significant improvement over Theorem 5.3. Other variants that "correct" the NM algorithm include [37, 104, 131]. Tseng's adaptations essentially consist of two safety checks. First, before evaluating a point, check that the potential simplex  $\tilde{\mathbb{Y}}$  satisfies

$$\text{von}(\tilde{\mathbb{Y}}) \geq \xi^v \quad \text{and} \quad \text{diam}(\tilde{\mathbb{Y}}) \leq \xi^e \text{ diam}(\mathbb{Y}^k).$$

If  $\tilde{\mathbb{Y}}$  does not satisfy this, then do not evaluate the point (as even if it were accepted the simplex would be unstable), instead, a simplex stability step is taken. The exact nature of the step depends on when the failure occurred:

- In a reflection step, the simplex stability step is a rotation,
- $$\mathbb{Y}^{k+1} = \mathbb{Y}^{rot} := \{y^0, y^0 - (y^1 - y^0), y^0 - (y^2 - y^0), \dots, y^0 - (y^n - y^0)\}.$$
- In an expansion step, the simplex stability accepts the rotation.
  - In a contraction step, the simplex stability step is a shrink.

The second safety check is to only accept contraction steps if "significant decrease occurs". In particular, the contraction value  $f^{cc} \in \{f^{ic}, f^{oc}\}$  needs to satisfy  $f^{cc} < f(y^n) - (\frac{1}{2} \text{diam}(\mathbb{Y}^k))^2$ . If it does not, then a shrink step is applied. With these two safety checks, any accumulation point of the vertices produced by the Fortified Nelder-Mead algorithm is guaranteed to have a null gradient, provided that  $f \in \mathcal{C}^1$ .

PART

# 3

## Direct Search Methods

Part 3 presents two families of direct search methods designed for black-box optimization problems.

- Chapter 6 introduces algebraic material on which the methods will rely, and elements from nonsmooth calculus required for rigorous and hierarchical convergence analyses.
- Chapter 7 presents the generalised pattern search algorithm, an evolution of the coordinate search algorithm that solves a wider class of optimization problems.
- Chapter 8 describes the mesh adaptive direct search algorithm. This method is supported by a comprehensive nonsmooth convergence analysis, and is the first algorithm of this book that can handle general inequality constraints.

Conclusion: direct search methods combine a flexible framework with proof of convergence.

# *Positive Bases and Nonsmooth Optimization*

Chapter 3 introduced a first practical DFO algorithm for unconstrained optimization, the coordinate search (CS) algorithm. While it was proven to converge to the first order in some circumstance (see Theorem 3.4), it was also noted that the algorithm can fail on very simple nondifferentiable convex functions (see Example 3.3). The CS algorithm is an example of the subclass of DFO methods called *direct search methods*. Direct search methods are methods that work from an *incumbent solution* and examine a collection of trial points. If improvement is found, then the incumbent solution is updated; while if no improvement is found, then a *step size* parameter is decreased and a new collection of trial points is examined.

In Part 3, we advance the basic ideas behind CS and provide two new direct search algorithms. The new algorithms, *generalised pattern search* (GPS) and *mesh adaptive direct search* (MADS), appear in Chapters 7 and 8 respectively. The methods generally differ from CS in three key aspects.

- i. First, the methods allow for searching in directions other than the  $2n$  positive and negative coordinate directions.
- ii. Second, the methods include a global search step in addition to the local poll step. This will allow for heuristic subroutines to be included within any practical implementation.
- iii. Third, in the event of a successful iteration, the methods allow for the search radius to increase.

Combined, these advances will allow for greater flexibility in implementation, widen the class of target problems, and provide stronger convergence analysis and improved numerical results.

In the current chapter, we provide some important background information required to present and understand the GPS and MADS algorithms. The first three sections present the geometrical notion of positive bases, and the last two sections discuss unconstrained and constrained optimality conditions for nonsmooth functions.

## 6.1. Positive Bases

In recalling the CS algorithm, we note that the core of the method was the poll step:

---

|  |  |
|--|--|
| 1. Poll  |  |
| if $f(t) < f(x^k)$ for some $t \in P^k := \{x^k \pm \delta^k e_i : i = 1, 2, \dots, n\}$ |  |
| set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \delta^k$                        |  |
| otherwise  |  |
| set $x^{k+1} \leftarrow x^k$ and $\delta^{k+1} \leftarrow \frac{1}{2}\delta^k$           |  |

---

Both the GPS and MADS algorithms will make an important modification to the poll step. Specifically, they will replace the set  $\{x^k \pm \delta^k e_i : i = 1, 2, \dots, n\}$ , with the more flexible set  $\{x^k + \delta^k d : d \in \mathbb{D}^k\}$ , where  $\mathbb{D}^k$  is some finite set of directions. In addition, these algorithms will make a second change by allowing the step size parameter  $\delta^k$  to increase in the event of a successful iteration, but we leave that to discuss in Chapter 7.

We shall call  $\mathbb{D}^k$  the set of *poll directions*, and the set  $P^k = \{x^k + \delta^k d : d \in \mathbb{D}^k\}$  the *polling set* at iteration  $k$ . It should be immediately clear that the success or failure of any direct search method is highly dependent on the polling set. Indeed, if we rotate the coordinate directions used by CS in Example 3.3 by any angle  $\theta \in (0, \pi/2)$ , then the resulting algorithm converges to the correct minimiser (see Exercise 6.1).

A good polling set should satisfy three conditions. First, it should contain vectors that point in every half-space of  $\mathbb{R}^n$ . Second, it should contain few vectors. Third, all vectors should be reasonably similar in length. The first two of these properties are mathematically captured in the definition of a *positive basis*. We begin by introducing the ideas of a *positive spanning set* and *positive linear independence*.

**DEFINITION 6.1 (Positive Spanning Set).**

Let  $\mathbb{D} \subseteq \mathbb{R}^n$ . The positive span of  $\mathbb{D}$ , denoted  $\text{pspan}(\mathbb{D})$ , is the set of all nonnegative linear combinations of vectors in  $\mathbb{D}$ :

$$\text{pspan}(\mathbb{D}) = \left\{ \sum_{i=1}^m \lambda_i d^i : \lambda_i \geq 0, d^i \in \mathbb{D}, m \in \mathbb{N} \right\} \subseteq \mathbb{R}^n.$$

The set  $\mathbb{D}$  is a positive spanning set for  $\mathbb{R}^n$  if and only if  $\text{pspan}(\mathbb{D}) = \mathbb{R}^n$ .

The above definition of pspan simplifies to

$$\text{pspan}(\mathbb{D}) = \left\{ \sum_{i=1}^{|\mathbb{D}|} \lambda_i d^i : \lambda_i \geq 0, d^i \in \mathbb{D} \right\} \subseteq \mathbb{R}^n$$

when  $\mathbb{D}$  is a finite set.

**DEFINITION 6.2 (Positive Linear Independence).**

A set  $\mathbb{D}$  is positive linearly independent if and only if  $d \notin \text{pspan}(\mathbb{D} \setminus \{d\})$  for all  $d \in \mathbb{D}$ .

The terms *positive span* and *positive linearly independent* are natural extensions of the terms *span* and *linearly independent* from Linear Algebra (see Section 2.1). To say a set is a positive spanning set means that the set contains enough information to recreate  $\mathbb{R}^n$  using nonnegative linear combinations. To say a set is positive linearly independent means that it does not contain redundant vectors: i.e., removing any one of its vectors reduces its positive span. If a set is both a positive spanning set and positive linearly independent, then we call it a *positive basis*.

**DEFINITION 6.3 (Positive Basis).**

A set  $\mathbb{D}$  is a positive basis of  $\mathbb{R}^n$  if and only if it is a positive spanning set of  $\mathbb{R}^n$  and is positive linearly independent.

**EXAMPLE 6.1.** Figure 6.1 illustrates four sets of vectors in  $\mathbb{R}^2$ . The set a) illustrates a *basis* of  $\mathbb{R}^2$  that is not a positive basis, as it is not a positive spanning set. Sets b), c), and d) are all positive spanning sets, but only b) and c) are positive bases. Set d) is not a positive basis as four of its vectors can be created as a positive combination of the remaining vectors.

This example raises an important point about positive bases. Unlike bases, the cardinality of a positive basis is not entirely determined by the dimension  $n$ . However, the dimension has some control over the cardinality of a positive basis.

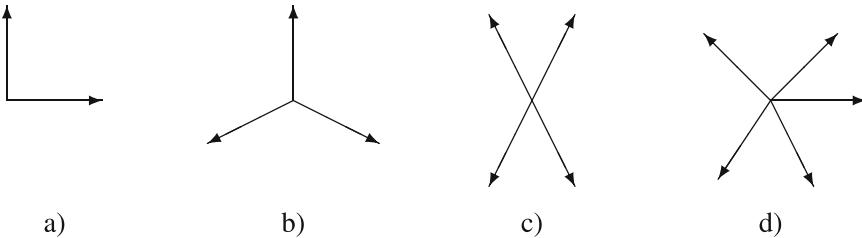


FIGURE 6.1. A basis, two positive bases, and a positive spanning set of  $\mathbb{R}^2$

Recall that if  $\mathbb{D}$  is a spanning set for  $\mathbb{R}^n$ , then  $\text{span}(\mathbb{D}) = \mathbb{R}^n$  and  $\mathbb{D}$  contains at least  $n$  vectors. Using this, we can easily acquire a lower bound on the size of a positive spanning set.

**COROLLARY 6.1 (Minimal Positive Spanning Set).**

*Let  $\mathbb{D}$  be a positive spanning set for  $\mathbb{R}^n$ . Then,  $\mathbb{D}$  contains at least  $n + 1$  vectors.*

**PROOF.** If  $\mathbb{D}$  contains infinitely many vectors, then the result is trivial, so we only consider the case of  $\mathbb{D} = \{d^1, d^2, \dots, d^m\}$ . Since  $\text{pspan}(\mathbb{D}) = \mathbb{R}^n$ , we know that there exists a nonnegative vector  $\lambda \in \mathbb{R}_+^m$  such that  $-d^1 = \sum_{i=1}^m \lambda_i d^i$ . Subtracting  $\lambda_1 d^1$  from both sides yields

$$\begin{aligned} -(1 + \lambda_1)d^1 &= \sum_{i=2}^m \lambda_i d^i \\ d^1 &= \sum_{i=2}^m \frac{-\lambda_i}{1 + \lambda_1} d^i \quad (\text{since } 1 + \lambda \neq 0). \end{aligned}$$

Thus  $d^1 \in \text{span}(\mathbb{D} \setminus \{d^1\})$ . From this we see

$$\text{span}(\mathbb{D} \setminus \{d^1\}) \supseteq \text{span}(\mathbb{D}) \supseteq \text{pspan}(\mathbb{D}) = \mathbb{R}^n,$$

so  $\mathbb{D} \setminus \{d^1\}$  is a spanning set for  $\mathbb{R}^n$ . Hence  $\mathbb{D} \setminus \{d^1\}$  contains at least  $n$  vectors, and therefore  $\mathbb{D}$  contains at least  $n + 1$  vectors.  $\square$

The upper bound on the size of a positive basis is trickier to obtain.

**THEOREM 6.2 (Maximal Positive Basis).**

*Let  $\mathbb{D}$  be a positive basis of  $\mathbb{R}^n$ . Then  $\mathbb{D}$  has at least  $n + 1$  vectors and at most  $2n$  elements.*

The lower bound in Theorem 6.2 follows immediately from Corollary 6.1. Several proofs showing the upper bound in Theorem 6.2 have been proposed. The longest of these is several pages, and the shortest of these relies heavily on theory from Linear Programming.<sup>1</sup> Interestingly, unlike a basis of  $\mathbb{R}^n$ , proving the upper bound on positive basis uses both properties that the set is positive spanning and positive linearly independent. (Recall, the upper bound on the cardinality of a basis comes from the fact that if  $\mathbb{D}$  is linearly independent, then  $\mathbb{D}$  contains at most  $n$  vectors.) On its own, positive linear independence does not provide any upper bound on the size of the set.

**EXAMPLE 6.2.** Denote by  $\lfloor \cdot \rfloor$  the function that rounds a real number down to the nearest integer, and denote by  $\lceil \cdot \rceil$  the function that rounds a real number up to the nearest integer. These are commonly called the *floor* and *ceiling* operators.

In  $\mathbb{R}^n$ , let  $\mathbb{D}$  be the set of vectors of all the nontrivial permutations with  $\lfloor \frac{n}{2} \rfloor$  1's and  $n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$  0's,

$$\mathbb{D} = \left\{ d \in \{0, 1\}^n : \sum_{i=1}^n d_i = \lfloor \frac{n}{2} \rfloor \right\}.$$

We claim that  $\mathbb{D}$  is positive linearly independent and contains  $C_n^{\lfloor \frac{n}{2} \rfloor} = \frac{n!}{(\lceil \frac{n}{2} \rceil)!(\lfloor \frac{n}{2} \rfloor)!}$  vectors.

Indeed, suppose  $d^k \in \mathbb{D}$  and there exist nonnegative numbers  $\lambda_i$  such that  $d^k = \sum_{i \neq k} \lambda_i d^i$  with  $d^i \in \mathbb{D}$  for every  $i$ . As  $d^k \neq 0$ , there must exist some  $j$  with  $\lambda_j > 0$ . Since  $d^k \neq d^j$ , and  $d^k, d^j \in \mathbb{D}$ , there must exist some index  $\ell$  with  $d_\ell^k = 0$  and  $d_\ell^j = 1$ . This yields the contradiction

$$0 = d_\ell^k = \sum_{i \neq k} \lambda_i d_\ell^i \geq \lambda_j d_\ell^j = \lambda_j > 0.$$

Hence,  $\mathbb{D}$  is positively linearly independent with  $C_n^{\lfloor \frac{n}{2} \rfloor}$  vectors.

## 6.2. Constructing Positive Bases

It is fairly easy to create examples of positive bases. In Figure 6.1, set b) is a positive basis with  $n + 1$  elements, and set c) is a positive basis with  $2n$  elements. This shows the two extremes of Theorem 6.2 can be obtained in  $\mathbb{R}^2$ . To see these extremes can be obtained in  $\mathbb{R}^n$  we consider two simple examples.

**EXAMPLE 6.3.** Let

$$\mathbb{D}_{\min} = \left\{ e_1, e_2, \dots, e_n, -\sum_{i=1}^n e_i \right\},$$

<sup>1</sup>A sketch of the proof appears in the further remarks section for Part 3, on page 154.

where  $e_i$  is the  $i$ -th coordinate vector. Then  $\mathbb{D}_{\min}$  is a *minimal positive basis* of  $\mathbb{R}^n$ , i.e., a positive basis of  $\mathbb{R}^n$  containing  $n + 1$  vectors. We call  $\mathbb{D}_{\min}$  the *canonical minimal positive basis* of  $\mathbb{R}^n$ .

Let

$$\mathbb{D}_{\max} = \{e_1, e_2, \dots, e_n, -e_1, -e_2, \dots, -e_n\}.$$

Then  $\mathbb{D}_{\max}$  is a *maximal positive basis* of  $\mathbb{R}^n$ , i.e., a positive basis of  $\mathbb{R}^n$  containing  $2n$  vectors. We call  $\mathbb{D}_{\max}$  the *canonical maximal positive basis* of  $\mathbb{R}^n$ .

Just as applying an invertible linear transform to a basis creates a new basis, applying an invertible linear transform to a positive basis creates a new positive basis.

**LEMMA 6.3 (Constructing Positive Bases).**

Let  $A \in \mathbb{R}^{n \times n}$  be an invertible matrix. Given a set  $\mathbb{D} = \{d^1, d^2, \dots, d^m\}$ , define  $\mathbb{B} = \{Ad^1, Ad^2, \dots, Ad^m\}$ .

- i. If  $\mathbb{D}$  is a positive spanning set, then  $\mathbb{B}$  is a positive spanning set.
- ii. If  $\mathbb{D}$  is positive linearly independent, then  $\mathbb{B}$  is positive linearly independent.
- iii. If  $\mathbb{D}$  is a positive basis of  $\mathbb{R}^n$ , then  $\mathbb{B}$  is a positive basis of  $\mathbb{R}^n$ .

**PROOF.** i. Suppose  $\mathbb{D}$  is a positive spanning set. Given any  $x \in \mathbb{R}^n$ , define  $\hat{x} = A^{-1}x$ . As  $\mathbb{D}$  is a positive spanning set, there exists  $\lambda \in \mathbb{R}_+^m$  such that  $\hat{x} = \sum_{i=1}^m \lambda_i d^i$ . Multiplying by  $A$  yields

$$x = \sum_{i=1}^m \lambda_i (Ad^i) \in \text{pspan}(\mathbb{B}).$$

Thus  $\mathbb{B}$  is a positive spanning set for  $\mathbb{R}^n$ .

ii. Suppose  $\mathbb{D}$  is positive linearly independent. For eventual contradiction, suppose there exist an index  $k$  such that

$$Ad^k \in \text{pspan}(\mathbb{B} \setminus \{Ad^k\}).$$

Then, there exists  $\lambda \in \mathbb{R}_+^m$  such that  $Ad^k = \sum_{i \neq k} \lambda_i Ad^i$ . Multiplying by  $A^{-1}$  gives

$$d^k = \sum_{i \neq k} \lambda_i A^{-1} Ad^i = \sum_{i \neq k} \lambda_i d^i.$$

This contradicts the fact that  $\mathbb{D}$  is positive linearly independent. Hence, such an index  $k$  does not exist, so  $\mathbb{B}$  is positive linearly independent.

iii. This follows immediately from parts i. and ii. □

Recall that given any basis  $\mathbb{B}$  of  $\mathbb{R}^n$ , there exists an invertible linear transform that maps  $\mathbb{B}$  to the canonical basis, i.e., there exists  $A \in \mathbb{R}^{n \times n}$  such that  $A\mathbb{B} = \{e_1, e_2, \dots, e_n\}$ . Combined with the above, this provides a method to create minimal and maximal positive bases from any basis in  $\mathbb{R}^n$ .

**THEOREM 6.4** (Minimal and Maximal Positive Bases).

Let  $\mathbb{B} = \{d^1, d^2, \dots, d^m\}$  be a basis of  $\mathbb{R}^n$  and

$$\begin{aligned}\mathbb{D}_{\min} &= \mathbb{B} \cup \left\{ -\sum_{i=1}^m d^i \right\}, \\ \mathbb{D}_{\max} &= \mathbb{B} \cup (-\mathbb{B}).\end{aligned}$$

Then,  $\mathbb{D}_{\min}$  is a minimal positive basis of  $\mathbb{R}^n$  and  $\mathbb{D}_{\max}$  is a maximal positive basis of  $\mathbb{R}^n$ .

**PROOF.** Apply Lemma 6.3 to Example 6.3. □

### 6.3. Positive Bases and Descent Directions

A very important property of positive spanning sets is that they contain at least one element in every open half-space of  $\mathbb{R}^n$ . Therefore, if  $f \in \mathcal{C}^1$  and  $\nabla f(x) \neq 0$ , then there is at least one element of any positive spanning set that is a descent direction (recall Definition 2.4).

**THEOREM 6.5** (Positive Bases and Descent Directions).

Let  $\mathbb{D}$  be a positive spanning set in  $\mathbb{R}^n$  and  $w$  a nonzero vector in  $\mathbb{R}^n$ . Then there exists  $d \in \mathbb{D}$  such that  $w^\top d < 0$ . Furthermore, if  $f \in \mathcal{C}^1$  and  $\nabla f(x) \neq 0$  for some  $x \in \mathbb{R}^n$ , then there exists  $d \in \mathbb{D}$  such that  $d$  is a descent direction of  $f$  at  $x$ .

**PROOF.** As  $\mathbb{D}$  is a positive spanning set with  $m$  elements, there exists  $\lambda \in \mathbb{R}_+^m$  such that

$$-w = \sum_{i=1}^m \lambda_i d^i, \text{ where } d^i \in \mathbb{D}.$$

Now, note that

$$0 > -\|w\|^2 = -w^\top w = \sum_{i=1}^m \lambda_i w^\top d^i.$$

Since  $\lambda_i \geq 0$  for all  $i$ , we must have  $w^\top d^i < 0$  for at least one index  $i$ .

To see the second statement, recall that  $f'(x; d) = \nabla f(x)^\top d$ . By the first statement of the theorem, if  $\nabla f(x) \neq 0$ , then there exists a vector  $d \in \mathbb{D}$  such that  $0 > \nabla f(x)^\top d = f'(x; d)$ , which implies that  $d$  is a descent direction for  $f$  at  $x$ . □

## 6.4. Optimality Conditions for Unconstrained Problems

We now turn our attention away from positive bases, and towards optimality conditions for nonsmooth functions. This will allow us to derive stronger and more general convergence analyses for the GPS and MADS algorithms presented in Chapters 7 and 8. In fact, despite the more general

convergence analyses, once we have the correct tools from nonsmooth optimisation, the proofs in Chapters 7 and 8 become quite easy.

We begin by generalising the notion of a directional derivative to the nonsmooth case. Recall that the directional derivative of a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  at  $x \in \mathbb{R}^n$  in the direction  $d \in \mathbb{R}^n$  is defined as

$$f'(x; d) = \lim_{t \searrow 0} \frac{f(x + td) - f(x)}{t},$$

when this limit exists. The notation  $t \searrow 0$  indicates that  $t$  goes to zero with positive values. If  $f \in \mathcal{C}^1$  near  $x$ , then there exists a unique vector  $\nabla f(x) \in \mathbb{R}^n$ , called the gradient of  $f$  at  $x$ , that satisfies  $f'(x; d) = \nabla f(x)^\top d$  for every  $d \in \mathbb{R}^n$ . If  $f \notin \mathcal{C}^1$ , then the directional derivative may still exist, as illustrated by the function  $f(x) = \|x\|$  whose directional derivative in the direction  $d \in \mathbb{R}$  at the point of nondifferentiability  $x = 0$  is  $f'(0; d) = \lim_{t \searrow 0} \frac{\|td\| - \|0\|}{t} = \|d\|$ . Thus, we immediately see that directional derivatives provide a more general structure than gradients from which to work.

The first order necessary optimality condition for the unconstrained minimisation of a differentiable function  $f \in \mathcal{C}^1$  can be written  $f'(x; d) \geq 0$  for all  $d \in \mathbb{R}^n$ . In fact, this statement does not require  $f \in \mathcal{C}^1$ , only that the directional derivatives exist at  $x$ .

**THEOREM 6.6 (First Order Optimality via  $f'$ ).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and  $\hat{x} \in \mathbb{R}^n$  be such that  $f'(\hat{x}; d)$  exists for all  $d \in \mathbb{R}^n$ . Then,

$$\text{if } \hat{x} \in \operatorname{argmin}_x \{f(x)\}, \text{ then } f'(\hat{x}; d) \geq 0 \text{ for all } d \in \mathbb{R}^n.$$

Moreover, if  $f$  is convex, then this statement becomes an “if and only if” statement

$$\text{if } f \text{ is convex and } f'(\hat{x}; d) \geq 0 \text{ for all } d \in \mathbb{R}^n, \text{ then } \hat{x} \in \operatorname{argmin}_x \{f(x)\}.$$

**PROOF.** Exercise 6.6. □

Unfortunately, the condition that  $f'(\hat{x}; d)$  exists for all  $d \in \mathbb{R}^n$  is a fairly strong assumption, and the convexity assumption is not compatible with BBO. Furthermore, working with directional derivatives would actually make the proofs in Chapters 7 and 8 fairly technical. Therefore, we now generalise the notion of a directional derivative to a definition that works nicely for any locally Lipschitz function. The following involves the notion of limit superior, from Definition 2.3.

**DEFINITION 6.4 (Generalised Directional Derivative).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be Lipschitz near  $x \in \mathbb{R}^n$ . The generalised directional derivative of  $f$  at  $x$  in the direction  $d \in \mathbb{R}^n$  is

$$f^\circ(x; d) = \limsup_{y \rightarrow x, t \searrow 0} \frac{f(y + td) - f(y)}{t}.$$

Another difference with the definition of the directional derivative is that the limit superior is taken by implicitly considering all sequences  $y$  that converge to  $x$ . The generalised directional derivative is also known as the *Clarke directional derivative*. Let us explore this definition with a simple single variable example.

**EXAMPLE 6.4.** Consider the function  $f(x) = |x|$ . For any  $d \in \mathbb{R}$ , the generalised directional derivative at  $x = 0$  is  $f^\circ(0; d) = |d|$ . Indeed

$$\begin{aligned} |d| &= f'(0; d) = \lim_{t \searrow 0} \frac{|td| - |0|}{t} \leq \limsup_{y \rightarrow 0, t \searrow 0} \frac{|y + td| - |y|}{t} \\ &= f^\circ(0; d) \\ &\leq \limsup_{y \rightarrow 0, t \searrow 0} \frac{|y| + |td| - |y|}{t} = |d|. \end{aligned}$$

Example 6.4 makes use of the fact that if the directional derivative exists, then, by definition of the  $\limsup$ ,

$$(6.1) \quad f'(x; d) = \lim_{t \searrow 0} \frac{f(x + td) - f(x)}{t} \leq \limsup_{y \rightarrow x, t \searrow 0} \frac{f(y + td) - f(y)}{t} = f^\circ(x; d).$$

The above inequality can be strict. For example, consider the function  $f(x) = \min\{x, 0\}$ : on the one hand, the directional derivative is  $f'(0; 1) = 0$  and on the other hand, the generalised directional derivative is  $f^\circ(0; 1) = 1$ .

Another very useful property of the generalised directional derivative is that whenever  $f^\circ(x; d)$  is well defined, it is bounded above through the Lipschitz constant of the function.

**PROPOSITION 6.7.**

Let  $f$  be Lipschitz continuous with constant  $K$  near  $x \in \mathbb{R}^n$ . Then  $f^\circ(x; d) \leq K\|d\|$  for any  $d \in \mathbb{R}^n$ .

**PROOF.** By definition

$$f^\circ(x; d) = \limsup_{y \rightarrow x, t \searrow 0} \frac{f(y + td) - f(y)}{t} \leq \limsup_{y \rightarrow x, t \searrow 0} \left| \frac{f(y + td) - f(y)}{t} \right|.$$

Applying the Lipschitz continuity of  $f$  yields

$$f^\circ(x; d) \leq \limsup_{y \rightarrow x, t \searrow 0} \frac{K\|y + td - y\|}{|t|} \leq \limsup_{y \rightarrow x, t \searrow 0} \frac{K\|td\|}{|t|} = K\|d\|.$$

□

Next, let us consider an example where the directional derivative  $f'$  does not exist, but the generalised directional derivative  $f^\circ$  does exist.

**EXAMPLE 6.5.** Consider the function

$$f(x) = \begin{cases} x \cos(\ln(|x|)) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0, \end{cases}$$

at the point  $\bar{x} = 0$ .

First, we show that the directional derivative  $f'(0; 1)$  is undefined, as

$$f'(0; 1) = \lim_{t \searrow 0} \frac{t \cos(\ln(|t|)) - 0}{t} = \lim_{t \searrow 0} \cos(\ln(|t|)),$$

which does not exist.

Second, we show that  $f$  is Lipschitz continuous. To see this, first consider  $f$  for  $x > 0$ . Note that  $f$  is differentiable and  $\nabla f(x) = \cos(\ln(x)) - x \sin(\ln(x)) \frac{1}{x} = \cos(\ln(x)) - \sin(\ln(x))$  when  $x > 0$ . An upper bound on  $|\nabla f(x)|$  follows from observing that for any angle  $\theta$

$$(\cos(\theta) - \sin(\theta))^2 = \cos(\theta)^2 + \sin(\theta)^2 - 2 \cos(\theta) \sin(\theta) = 1 - \sin(2\theta) \leq 2.$$

Thus  $|\nabla f(x)| \leq \sqrt{2}$  for all  $x > 0$ , which implies  $f$  is Lipschitz with constant  $K = \sqrt{2}$  for  $x > 0$ . Similar arguments show that  $f$  is Lipschitz with constant  $\sqrt{2}$  at any point  $x \neq 0$ . At the point  $\bar{x} = 0$  we have  $|f(y) - f(0)| = |y \cos(\ln(|y|))| \leq |y|$ , which implies  $f$  is Lipschitz with constant 1 at  $\bar{x} = 0$ .

Thus we have shown that  $f$  is everywhere Lipschitz continuous with constant  $\sqrt{2}$ . Finally, Proposition 6.7 ensures that  $f^\circ(0; 1)$  is well defined (as  $f$  is Lipschitz) and  $f^\circ(0; 1) \leq \sqrt{2}$ .

It turns out that if  $f$  is Lipschitz and convex near  $x$ , then the directional derivatives and the generalised directional derivatives are identical. This is also true of any  $\mathcal{C}^1$  function. When the directional derivatives and the generalised directional derivatives coincide we call the function *regular*.

**DEFINITION 6.5 (Regular Function).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be Lipschitz near  $x \in \mathbb{R}^n$ . The function  $f$  is said to be regular at  $x$  if for every directions  $d \in \mathbb{R}^n$ , the directional derivative  $f'(x; d)$  exists and equals  $f^\circ(x; d)$ .

**THEOREM 6.8** (Convex and  $\mathcal{C}^1$  Functions Are Regular).

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ .

- i. If  $f \in \mathcal{C}^1$ , then  $f$  is regular at all  $x \in \mathbb{R}^n$ .
- ii. If  $f$  is Lipschitz and convex near  $x \in \mathbb{R}^n$ , then  $f$  is regular at  $x$ .

**PROOF.** i. Suppose  $f \in \mathcal{C}^1$ . Recall that Definition 2.1 ensures that given any  $x \in \mathbb{R}^n$ , the gradient  $\nabla f(x)$  exists and is the unique vector satisfying

$$\lim_{u \rightarrow x} \frac{f(u) - f(x) - \nabla f(x)^\top (u - x)}{\|u - x\|} = 0.$$

In particular, for any  $d \in \mathbb{R}^n$

$$\lim_{t \searrow 0} \frac{f(x + td) - f(x) - \nabla f(x)^\top (td)}{t \|d\|} = 0.$$

Since  $\nabla f$  is continuous, this further implies that

$$\lim_{y \rightarrow x} \lim_{t \searrow 0} \frac{f(y + td) - f(y) - \nabla f(y)^\top (td)}{t} = 0.$$

Furthermore,  $f \in \mathcal{C}^1$  implies that  $f$  is locally Lipschitz continuous, so  $f^\circ$  is well defined. Thus

$$\begin{aligned} f^\circ(x; d) &= \limsup_{y \rightarrow x, t \searrow 0} \frac{f(y + td) - f(y) - \nabla f(y)^\top (td) + \nabla f(y)^\top (td)}{t} \\ &= 0 + \limsup_{y \rightarrow x, t \searrow 0} \frac{\nabla f(y)^\top (td)}{t} = \nabla f(x)^\top d = f'(x; d) \end{aligned}$$

which implies that  $f$  is a regular function.

ii. Suppose  $f$  is Lipschitz and convex near  $x \in \mathbb{R}^n$ . Select  $d \in \mathbb{R}^n$  and let  $K$  be the Lipschitz constant of  $f$ . Let  $\delta$  be any scalar strictly greater than 0, and consider any sequences  $y^k \rightarrow x$  and  $t^k \searrow 0$  and define  $\epsilon^k = \max\{t^k, \frac{1}{\delta} \|y^k - x\|\} > 0$ . This implies that for any  $k$

$$\begin{aligned} \frac{f(y^k + t^k d) - f(y^k)}{t^k} &\leq \frac{f(y^k + \epsilon^k d) - f(y^k)}{\epsilon^k} \\ &\quad (\text{by convexity of } f \text{ and Exercise 6.8}) \\ &= \frac{f(y^k + \epsilon^k d) - f(x + \epsilon^k d) + f(x + \epsilon^k d) - f(x) + f(x) - f(y^k)}{\epsilon^k} \\ &\leq \frac{f(x + \epsilon^k d) - f(x) + 2K \|y^k - x\|}{\epsilon^k} \\ &\leq \frac{f(x + \epsilon^k d) - f(x)}{\epsilon^k} + 2K\delta. \end{aligned}$$

Now, since  $\epsilon \searrow 0$ ,  $\|y - x\| \leq \delta\epsilon$ ,  $0 \leq t \leq \epsilon$  implies  $y \rightarrow x$ ,  $t \searrow 0$ , the generalised directional derivative satisfies

$$\begin{aligned} f^\circ(x; d) &= \limsup_{y \rightarrow x, t \searrow 0} \frac{f(y + td) - f(y)}{t} \\ &\leq \limsup_{\epsilon \searrow 0, \|y-x\| \leq \delta\epsilon, 0 \leq t \leq \epsilon} \frac{f(y + td) - f(y)}{t} \\ &\leq \limsup_{\epsilon \searrow 0} \frac{f(x + \epsilon d) - f(x)}{\epsilon} + 2K\delta = f'(x; d) + 2K\delta. \end{aligned}$$

Applying Equation (6.1) we have  $f'(x; d) \leq f^\circ(x; d) \leq f'(x; d) + 2K\delta$  for any  $\delta > 0$ . As  $\delta > 0$  was arbitrary, we must have  $f^\circ(x; d) = f'(x; d)$ .  $\square$

At this point we have a number of local properties that a function can possess: continuous, locally Lipschitz (Definition 2.2), differentiable at a point (Definition 2.1), regular (Definition 6.5),  $\mathcal{C}^1$  (page 17), and convex (Definition 2.13). Figure 6.2 summarises the hierarchy of these properties. Note the importance of examining *locally* Lipschitz, as there are convex functions that are not globally Lipschitz (e.g.,  $f(x) = x^2$ ).

With a good understanding of generalised directional derivatives now established, we present first order optimality conditions in terms of the generalised directional derivatives.

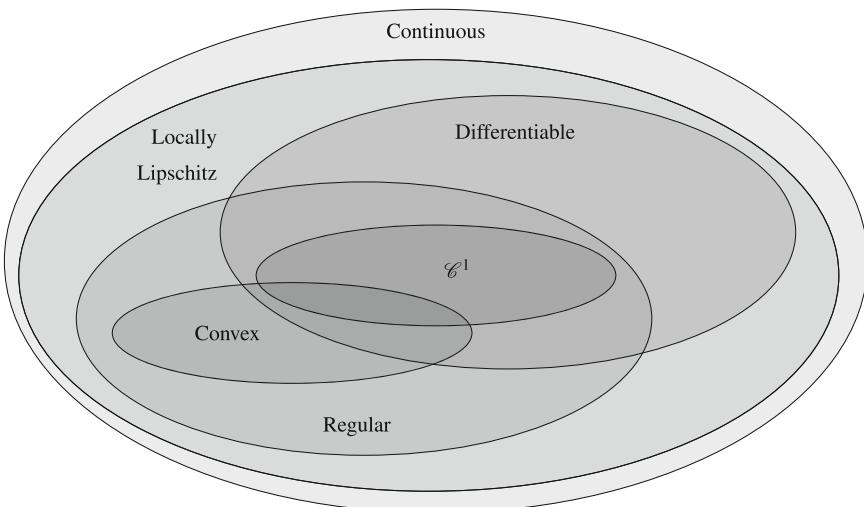


FIGURE 6.2. Local properties of continuous functions at or near some point  $x \in \mathbb{R}^n$

**THEOREM 6.9 (First Order Optimality via  $f^\circ$ ).**

If  $f$  is Lipschitz continuous near  $x^*$ , a local minimiser of  $f$ , then  $f^\circ(x^*; d) \geq 0$  for every  $d \in \mathbb{R}^n$ .

**PROOF.** Suppose that there is a direction  $d \in \mathbb{R}^n$  such that  $f^\circ(x^*; d) < 0$ . Then for every  $y$  sufficiently close to  $x^*$  and  $t > 0$  sufficiently close to 0, we would have that  $f(y + td) - f(y) < 0$ . In particular, setting  $y = x^*$  yields  $f(x^* + td) < f(x^*)$ , which contradicts the fact that  $x^*$  is a local minimiser of  $f$ .  $\square$

Note that Theorem 6.6 now follows easily from Theorems 6.8 and 6.9. The next example illustrates Theorem 6.9 on a function with two variables.

**EXAMPLE 6.6.** Consider the function  $f(x) = |x_1 x_2|$  and  $x^* = (0, 0)^\top$ . The function is nonnegative, and therefore  $x^*$  is a global minimiser, and therefore the generalised directional derivative of  $f$  at  $x^*$  in any direction  $d \in \mathbb{R}^2$  is nonnegative. Indeed,

$$\begin{aligned} f^\circ(x^*; d) &= \limsup_{y \rightarrow 0, t \searrow 0} \frac{|(y_1 + td_1)(y_2 + td_2)| - |y_1 y_2|}{t} \\ &\geq \lim_{t \searrow 0, y=td} \frac{|(y_1 + td_1)(y_2 + td_2)| - |y_1 y_2|}{t} \\ &= \lim_{t \searrow 0} \frac{|4t^2 d_1 d_2| - |t^2 d_1 d_2|}{t} \\ &= \lim_{t \searrow 0} |3td_1 d_2| = 0. \end{aligned}$$

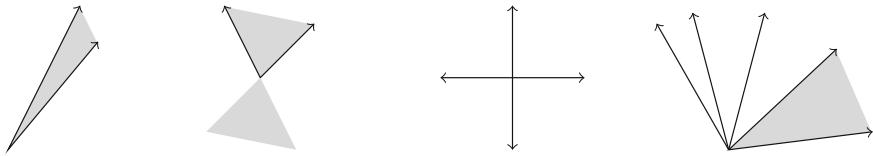
## 6.5. Optimality Conditions for Constrained Problems

Theorem 6.9 covers the situation of unconstrained optimization problems. To understand optimality conditions for constrained optimization problems in the context of direct-search methods, we must introduce the notion of cones, and specifically the *hypertangent cone*.

**DEFINITION 6.6 (Cone).**

A set  $T \subseteq \mathbb{R}^n$  is said to be a cone if and only if  $\lambda d \in T$  for every scalar  $\lambda > 0$  and for every  $d \in T$ .

The empty set, the singleton  $\{0\} \subset \mathbb{R}^n$ , and the entire space  $\mathbb{R}^n$  are all cones. A cone can be open or closed, convex or not. Some more examples of cones appear in Figure 6.3.

FIGURE 6.3. Four examples of cones in  $\mathbb{R}^2$ 

In the presence of constraints, the optimality condition of Theorem 6.9 needs some adjustments. Consider the constrained optimization problem

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

where  $\Omega \subseteq \mathbb{R}^n$ . In situations where the functions that define the set  $\Omega$  are differentiable, and when the gradients of these functions are available, one can use them to construct the tangent cone at  $x$ , i.e., the set of directions in  $\mathbb{R}^n$  that point inside  $\Omega$ . Unfortunately, in a DFO setting these gradients are unavailable, or may simply not exist. This forces us to use a more complicated and more general definition of tangency. In order to work with this DFO problem, we define the *hypertangent cone*.

**DEFINITION 6.7 (Hypertangent Cone).**

A vector  $d \in \mathbb{R}^n$  is said to be a hypertangent vector to the set  $\Omega \subset \mathbb{R}^n$  at the point  $x \in \Omega$  if and only if there exists a scalar  $\epsilon > 0$  such that (6.2)

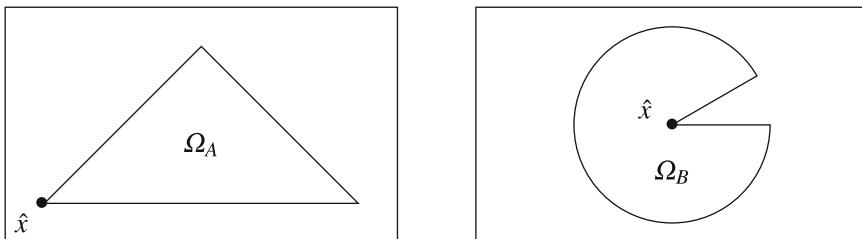
$$y + tw \in \Omega \quad \text{for all } y \in \Omega \cap B_\epsilon(x), \quad w \in B_\epsilon(d) \quad \text{and } 0 < t < \epsilon.$$

The set of all hypertangent vectors to  $\Omega$  at  $x$  is called the hypertangent cone to  $\Omega$  at  $x$ , and is denoted by  $T_\Omega^H(x)$ .

The hypertangent cone is always an open and convex set (Exercise 6.9). The next example helps us understand the hypertangent cone.

**EXAMPLE 6.7.** Consider the two sets illustrated in Figure 6.4:

$$\begin{aligned} \Omega_A &= \{(a, b)^T : 0 \leq b \leq a, a + b \leq 1\} \quad \text{and} \\ \Omega_B &= \{(a, b)^T : a^2 + b^2 \leq 1\} \setminus \{((a, b)^T : 0 < 2b < a\}. \end{aligned}$$

FIGURE 6.4. The set  $\Omega_A$  and  $\Omega_B$  from Example 6.7

The hypertangent cone of  $\Omega_A$  at  $\hat{x}$  is exactly what we would expect, the set of directions that point strictly into the set. Notice,  $T_{\Omega_A}^H(0)$  is open, and therefore does not include the boundary directions  $[1, 0]^\top$  and  $[1, 1]^\top$ .

The hypertangent cone of  $\Omega_B$  is more subtle. For a direction  $d$  to be a hypertangent vector of  $\Omega$  at  $\hat{x}$ , we need that

given any point  $y \in \Omega_B$  near  $\hat{x}$  along with any direction  $w \in \mathbb{R}^n$  near  $d$ ,

to be able to

select a step length  $\epsilon$  such that  $y + tw \in \Omega_B$  for all  $0 < t < \epsilon$ .

Starting at  $\hat{x} = 0 \in \Omega_B$ , if we consider the point  $y = [\delta, 0]^\top \in \Omega_B$  with  $\delta > 0$ , then any vector pointing upwards is not in  $\Omega_B$ . Moreover, the vector  $[-1, 0]^\top \notin T_{\Omega_B}^H(0)$ , as we would require  $[\delta, 0]^\top + t[-1, \gamma]^\top$  to be in  $\Omega_B$  for all  $\gamma \in \mathbb{R}$  and  $t > 0$  sufficiently small, but  $[\delta, 0]^\top + t[-1, \gamma]^\top \notin \Omega_B$  whenever  $t < \frac{\delta}{2}$  and  $0 < \gamma < \frac{\delta-t}{2t}$ .

The resulting hypertangent cones of  $\Omega_A$  and  $\Omega_B$  are

$$T_{\Omega_A}^H(0) = \{(a, b)^T : 0 < b < a\} \quad \text{and} \quad T_{\Omega_B}^H(0) = \{(a, b)^T : b < 0 \text{ and } 2b > a\}.$$

Notice the *and* instead of an *or* in the hypertangent cone  $T_{\Omega_B}^H(0)$ .

In Figure 6.5 we illustrate the sets  $T_{\Omega_A}^H(0)$  and  $T_{\Omega_B}^H(0)$  superimposed on top of  $\Omega_A$  and  $\Omega_B$ .

We now present two types of necessary optimality conditions based on the tangent cone definitions.

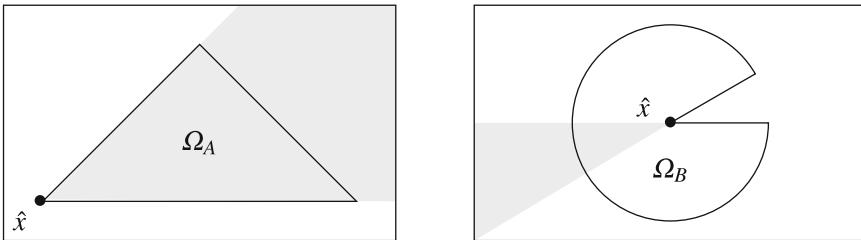


FIGURE 6.5. The hypertangent cones of  $\Omega_A$  and  $\Omega_B$  from Example 6.7

**THEOREM 6.10 (Constrained First Order Optimality via  $f^\circ$ ).**

If  $f$  is Lipschitz continuous near  $x^*$ , a local minimiser of  $f$  over the constraint set  $\Omega$ ,

$$x^* \in \operatorname{argmin}_x \{f(x) : x \in \Omega\},$$

then  $f^\circ(x^*; d) \geq 0$  for every direction  $d \in T_\Omega^H(x^*)$ .

**PROOF.** Suppose by contradiction that there is an hypertangent direction  $d \in T_{\Omega}^H(x^*)$  such that  $f^\circ(x^*; d) < 0$ . Then for every  $y \in \Omega$  sufficiently close to  $x^*$  and  $t > 0$  sufficiently close to 0, we would have that  $y + td \in \Omega$  and  $f(y + td) - f(y) < 0$ . In particular, setting  $y = x^*$  yields  $f(x^* + td) < f(x^*)$ , which contradicts the fact that  $x^*$  is a local minimiser of  $f$ .  $\square$

If  $f^\circ(\hat{x}; d) \geq 0$  for every direction  $d \in T_{\Omega}^H(\hat{x})$ , then we call  $\hat{x}$  a *hypertangent stationary point* of  $f$  over  $\Omega$ .

The next theorem provides a formula for the hypertangent cone when  $\Omega$  is given by differentiable functions.

**THEOREM 6.11 (Hypertangent Cone for Smooth Functions).**

Suppose  $\Omega = \{x : c_j(x) \leq 0, j \in J\}$  where  $c_j \in \mathcal{C}^1$ . Suppose  $x \in \Omega$  is a point such that  $\{\nabla c_j(x) : c_j(x) = 0\}$  is linearly independent. Then the hypertangent cone is

$$T_{\Omega}^H = \{d : \nabla c_j(x)^\top d < 0 \text{ for all } j \in J \text{ such that } c_j(x) = 0\}.$$

The proof of Theorem 6.11 is actually quite hard and requires a clever application of the inverse function theorem. For this book, we shall simply accept it as fact.

**EXAMPLE 6.8.** Figure 6.6 illustrates the above optimality conditions on  $\mathcal{C}^1$  functions in  $\mathbb{R}^2$ . Level sets of the objective function  $f$  are represented by the thin curves, and the domain  $\Omega$  is the region delimited by the two thick curves  $c_1(x) = 0$  and  $c_2(x) = 0$ . Let us discuss the three local minimisers  $x_a, x_b$ , and  $x_c$  of  $f$  on  $\Omega$ .

- i. At  $x_a$ , there are no active constraints (i.e.,  $c_j(x) < 0$  for  $j \in \{1, 2\}$ ). As such, the hypertangent cone is  $\mathbb{R}^2$ , which is represented by the shaded circle. All directional derivatives of  $f$  at  $x_a$  are null because  $\nabla f(x_a) = 0$ .
- ii. At  $x_b$ , only the constraint  $c_1$  is active, and the hypertangent cone is the half-space, opposite to the gradient of  $c_1$ , represented by the shaded half circle. The directional derivatives of  $f$  are nonnegative in this half space.
- iii. Both constraints  $c_1$  and  $c_2$  are active at  $x_c$ . The hypertangent cone is composed of the directions that make a negative inner product with the gradient of both constraints. It is represented by the shaded conic region, and the directional derivative of  $f$  are positive in this cone.

Any other point in  $\Omega$  has an hypertangent direction which is a descent direction for the objective function  $f$ , and thus, is not a minimiser.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

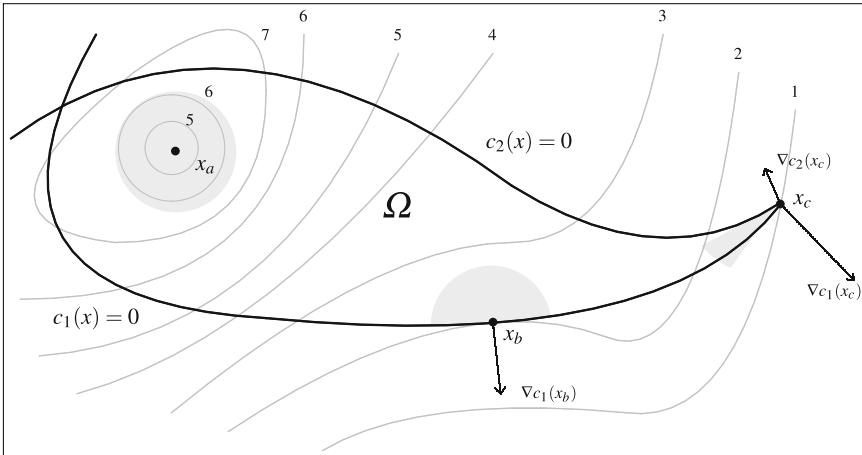


FIGURE 6.6. Necessary optimality conditions for constrained optimization

**EXERCISE 6.1.**  $\square$  Recall Example 3.3, where we saw that the CS algorithm applied to  $f(x) = \max\{|x_1|, |x_2|\}$  starting at  $[1, 1]^\top$  failed to converge to the global minimiser.

- Prove  $f$  is a convex function.
- Let  $\theta \in (0, \pi/2)$  be given, and define the rotation matrix

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}.$$

Prove  $R_\theta$  is invertible.

- Randomly select  $\theta$  in  $(0, \pi/2)$  and define  $g(x) = f(R_\theta x)$ . Run 10 iterations the CS algorithm with complete polling to minimise  $g$  starting at the point  $[1, 1]^\top$ .

Repeat the above for a total of 5 values of  $\theta$ . Provide a table of the results.

**EXERCISE 6.2.** + Recall Example 3.3, where we saw that CS applied to  $f(x) = \max\{|x_1|, |x_2|\}$  starting at  $[1, 1]^\top$  failed to converge to the global minimiser. Let  $\theta \in (0, \pi/2)$  be given, define the rotation matrix  $R_\theta$  and the function  $g(x) = f(R_\theta x)$  as in Exercise 6.1.

- Suppose the CS algorithm is applied to minimise  $g$  starting at the point  $R_\theta[1, 1]^\top$ . Explain why this is equivalent to rotating the coordinate directions used by CS.
- Suppose the CS algorithm is applied to minimise  $g$  starting at the point  $R_\theta[1, 1]^\top$ . Prove that the algorithm will converge to the true minimiser  $[0, 0]^\top$ .

[Hint: use the fact  $\mathcal{L}_g(c)$  is compact and apply Theorem 3.4.]

**EXERCISE 6.3.** Create a positive basis of  $\mathbb{R}^3$  with:

- a) Exactly 6 vectors.
- b) Exactly 5 vectors.
- c) Exactly 4 vectors.
- +d) Exactly 4 vectors, and the angle between any two of them is identical.

**EXERCISE 6.4.** + Consider the set  $\Omega = \{(x, y, z) : x^2 + y^2 = 1, z = 1\}$ . Let  $\mathbb{D}$  be any finite subset of  $\Omega$ :  $\mathbb{D} = \{v^1, v^2, \dots, v^k\} \subset \Omega$ .

- a) Prove that  $\mathbb{D}$  is positive linearly independent.
- b) Prove that  $\Omega$  is positive linearly independent.

**EXERCISE 6.5.** Prove Theorem 6.4.

**EXERCISE 6.6.** + Prove Theorem 6.6.

**EXERCISE 6.7.** Consider the function from Example 6.5

$$f(x) = \begin{cases} x \cos(\log(|x|)) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0. \end{cases}$$

- a) Create two sequences  $\{s^k\}$  and  $\{t^k\}$  such that  $s^k \searrow 0, t^k \searrow 0$ ,

$$\lim_{k \rightarrow \infty} \frac{s^k \cos(\log(|s^k|)) - 0}{s^k} = -1,$$

and

$$\lim_{k \rightarrow \infty} \frac{t^k \cos(\log(|t^k|)) - 0}{t^k} = 1.$$

- b) Explain why part a) proves that  $f'(0; 1)$  does not exist.
- c) Find a sequence  $\{u^k\}$  such that  $u^k \searrow 0$  and  $\nabla f(u^k) = \sqrt{2}$  to show that  $f^\circ(0; 1) = \sqrt{2}$ .

[Hint: Example 6.5 gives  $f^\circ(0; 1) \leq \sqrt{2}$ .]

**EXERCISE 6.8.** + Suppose  $f : \mathbb{R}^n \mapsto \mathbb{R}$  is convex. Prove that for any  $\epsilon > 0$

$$\max_{0 < \tau \leq \epsilon} \frac{f(y + \tau d) - f(y)}{\tau} = \frac{f(y + \epsilon d) - f(y)}{\epsilon}.$$

**EXERCISE 6.9.** + Let  $\Omega \subseteq \mathbb{R}^n$  and  $x \in \Omega$ . Prove that the hypertangent cone  $T_\Omega^H(x)$  is

- a) Open.
- b) Convex.

**EXERCISE 6.10.** + Provide details proving that the hypertangent cone of  $\Omega_A = \{[a, b]^\top : 0 \leq b \leq a, a + b \leq 1\}$  at  $[0, 0]^\top$  is  $T_{\Omega_A}^H(0) = \{[a, b]^\top : 0 < b < a\}$ .

**EXERCISE 6.11.** + Provide details proving that the hypertangent cone of  $\Omega_B = \{[a, b]^\top : a^2 + b^2 \leq 1\} \setminus \{[a, b]^\top : 0 < 2b < a\}$  at  $[0, 0]^\top$  is  $T_{\Omega_B}^H(0) = \{[a, b]^\top : b < 0 \text{ and } 2b > a\}$ .

**EXERCISE 6.12.** Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be Lipschitz with constant  $K$ . Show the following properties of the generalised directional derivatives using Definition 6.4.

- a)  $f^\circ(x; \lambda d) = \lambda f^\circ(x; d)$  for every scalar  $\lambda > 0$  and all  $d \in \mathbb{R}^n$ .
- b)  $f^\circ(x; u + v) \leq f^\circ(x; u) + f^\circ(x; v)$  for every  $u, v \in \mathbb{R}^n$ .
- c)  $f^\circ(x; -d) = (-f)^\circ(x; d)$  for every  $d \in \mathbb{R}^n$ .
- +d)  $d \mapsto f^\circ(x; d)$  is a convex function with respect to  $d$ .

**EXERCISE 6.13.** +

- a) Propose a Lipschitz function from  $\mathbb{R}^2$  to  $\mathbb{R}$  such that at  $\bar{x} = [0, 0]^\top$

$$f^\circ(\bar{x}; d) = \begin{cases} d_1 + d_2 & \text{if } d_1 \leq 0 \text{ and } d_2 \leq 0 \\ -2d_1 + d_2 & \text{if } 0 < d_1 \text{ and } d_1 \geq d_2 \\ d_1 - 2d_2 & \text{if } 0 < d_2 \text{ and } d_2 > d_1. \end{cases}$$

- b) Show that there are no Lipschitz functions from  $\mathbb{R}^2$  to  $\mathbb{R}$  such that at  $\bar{x} = [0, 0]^\top$

$$f^\circ(\bar{x}; d) = \begin{cases} d_1 + d_2 & \text{if } d_1 \geq 0 \text{ and } d_2 \geq 0 \\ d_1 + 2d_2 & \text{if } d_1 \geq d_2 \text{ and } d_2 < 0 \\ 2d_1 + d_2 & \text{if } d_2 > d_1 \text{ and } d_1 < 0. \end{cases}$$

[Hint: use a property from Exercise 6.12.]

**EXERCISE 6.14.** Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  be Lipschitz near  $\bar{x} \in \mathbb{R}^n$ , and let  $d$  be some nonzero direction in  $\mathbb{R}^n$ .

- a) Use the fact that  $f$  is Lipschitz near  $\bar{x}$  to show that

$$\limsup_{y \rightarrow \bar{x}, w \rightarrow d, t \searrow 0} \frac{f(y+tw) - f(y)}{t} \leq f^\circ(\bar{x}; d).$$

- b) Set  $w = d$  to show that

$$\limsup_{y \rightarrow \bar{x}, w \rightarrow d, t \searrow 0} \frac{f(y+tw) - f(y)}{t} \geq f^\circ(\bar{x}; d).$$

- c) Conclude that

$$f^\circ(\bar{x}; d) = \limsup_{y \rightarrow \bar{x}, w \rightarrow d, t \searrow 0} \frac{f(y+tw) - f(y)}{t}.$$

- d) Prove that in order to show that the generalised directional derivative is nonnegative at  $\bar{x}$  in the direction  $d$ , it suffices to generate three subsequences:  $\{y_k\}$  converging to  $\bar{x}$ ,  $\{w_k\}$  converging to  $d$ , and  $\{t_k\}$  converging to zero from above in such a way that  $f(y_k) \leq f(y_k + t_k w_k)$  for infinitely many  $k$ 's.



**Chapter 6 Project: Explore the Clarke Subdifferential.** The goal of this project is to explore the Clarke subdifferential, also known as the *generalised gradient*, in particular its relation to optimality conditions.

**DEFINITION 6.8 (Clarke Subdifferential).**

Let  $f$  be a Lipschitz function. For any  $x \in \mathbb{R}^n$  define the Clarke subdifferential as the following set

$$\partial f(x) := \{v \in \mathbb{R}^n : f^\circ(x; d) \leq d^\top v, \text{ for all } d \in \mathbb{R}^n\}.$$

Graph each of the following functions and determine  $\partial f(x)$  at the point  $\bar{x}$ .

- a)  $f(x) = |x|$  ( $x \in \mathbb{R}^1$ ) at  $\bar{x} = 0$ .
- b)  $f(x) = \|x\|$  ( $x \in \mathbb{R}^2$ ) at  $\bar{x} = [0, 0]^\top$ .
- c)  $f(x) = (x_1)^2 + (x_2)^2$  at  $\bar{x} = [1, 0]^\top$ .
- d)  $f(x) = |x_1| + (x_2)^2$  at  $\bar{x} = [0, 1]^\top$ .
- e)  $f(x) = \max\{x_1, x_2\}$  at  $\bar{x} = [1, 1]^\top$ .

Prove the following statements.

- f) If  $f \in \mathcal{C}^1$ , then  $\partial f(x) = \{\nabla f(x)\}$ .
- g) Given any Lipschitz function  $f$ , the generalised directional derivative may be obtained via

$$f^\circ(x; d) = \max_v \{d^\top v : v \in \partial f(x)\}.$$

- h) If  $\bar{x} \in \operatorname{argmin}_{x \in \mathbb{R}^n} \{f(x)\}$ , then  $0 \in \partial f(\bar{x})$ .
- i) If  $f$  is regular and  $0 \notin \partial f(\bar{x})$ , then  $-\operatorname{Proj}(0, \partial f(\bar{x}))$  is a descent direction, where

$$\operatorname{Proj}(0, \partial f(\bar{x})) := \operatorname{argmin}_y \{\|y - \bar{x}\| : y \in \partial f(\bar{x})\}$$

is the projection of 0 onto the set  $\partial f(\bar{x})$ .

- j) If  $f$  is convex and  $0 \in \partial f(\bar{x})$ , then  $\bar{x} \in \operatorname{argmin}_x \{f(x)\}$ .

# *Generalised Pattern Search*

As mentioned in Chapter 6, the *generalised pattern search* (GPS) algorithm is an advancement on the CS algorithm first introduced in Chapter 3. The GPS algorithm follows the same basic logic behind the CS algorithm, namely it iteratively searches a collection of points seeking improvement over the incumbent solution. In order to increase the flexibility in how those points are selected, the GPS algorithm replaces the coordinate directions with a more general collection of vectors. Moreover, it will allow for these search vectors to change from iteration to iteration, and allow for evaluations that are not based on polling around the incumbent solution. The local exploration near the incumbent solution is called the *poll* step, and the more flexible exploration in the space of variables is called the *search* step.

To ensure convergence we will still need some control over the search and poll steps. This control will be linked to the notion of a *mesh*, an enumerable discretisation of the space of variables.

**DEFINITION 7.1 (Mesh).**

Let  $G \in \mathbb{R}^{n \times n}$  be invertible and  $Z \in \mathbb{Z}^{n \times p}$  be such that the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^n$ . Define  $D = GZ$ . The mesh generated by  $D$  centred at the incumbent solution  $x^k \in \mathbb{R}^n$  of coarseness  $\delta^k > 0$  is defined by

$$M^k := \{x^k + \delta^k Dy : y \in \mathbb{N}^p\} \subset \mathbb{R}^n.$$

Notice that since the columns of the matrix  $Z \in \mathbb{Z}^{n \times p}$  form a positive spanning set for  $\mathbb{R}^n$ , and since  $G \in \mathbb{R}^{n \times n}$  is invertible, then Lemma 6.3 ensures that the columns of  $D = GZ$  form a positive spanning set for  $\mathbb{R}^n$ . We say that  $D$  is a positive spanning matrix.

Let  $\mathbb{D}$  denote the set of directions in  $\mathbb{R}^n$  composed of the columns of  $D$ . Think of  $\mathbb{D}$  as being the set of possible directions in which the algorithm will explore the space of variables. In CS,  $\mathbb{D}$  was simply composed of the positive and negative coordinate directions. GPS allows  $\mathbb{D}$  to contain a wider variety of directions.<sup>1</sup>

## 7.1. The GPS Algorithm

We are now ready to present the GPS algorithm for unconstrained minimisation,

$$\min_x \{f(x) : x \in \mathbb{R}^n\}.$$

**ALGORITHM 7.1. Generalised pattern search (GPS)**

Given  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and starting point  $x^0 \in \mathbb{R}^n$

## 0. Initialisation

|  |                                |
|--|--------------------------------|
| $\delta^0 \in (0, \infty)$               | initial mesh size parameter    |
| $D = GZ$                                 | positive spanning matrix       |
| $\tau \in (0, 1)$ , with $\tau$ rational | mesh size adjustment parameter |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance             |
| $k \leftarrow 0$                         | iteration counter              |

## 1. Search

|  |
|--|
| if $f(t) < f(x^k)$ for some $t$ in a finite subset $S^k$ of the mesh $M^k$             |
| set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$ and go to 3 |
| otherwise go to 2  |

## 2. Poll

|   |
|---|
| select a positive spanning set $\mathbb{D}^k \subseteq \mathbb{D}$                  |
| if $f(t) < f(x^k)$ for some $t \in P^k = \{x^k + \delta^k d : d \in \mathbb{D}^k\}$ |
| set $x^{k+1} \leftarrow t$ and $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$          |

otherwise  $x^k$  is a mesh local optimizer  
set  $x^{k+1} \leftarrow x^k$  and  $\delta^{k+1} \leftarrow \tau\delta^k$

<sup>1</sup>A similar distinction was made in Section 2.6 when distinguishing a simplex  $\mathbb{Y}$  with the matrix  $Y$  formed by vertices.

### 3. Termination

```

    | if  $\delta^{k+1} \geq \epsilon_{\text{stop}}$ 
    |   increment  $k \leftarrow k + 1$  and go to 1
    | otherwise stop
  
```

---

Each iteration of the **GPS** algorithm is divided into two main steps. First, the **search** step is free to apply any strategy to select candidate mesh points at which  $f$  will be evaluated, provided that only a finite number of mesh points (possibly none) are selected. If the **search** step fails in finding a feasible improved mesh point, then the **poll** step must be invoked. At this stage the function is evaluated at neighbouring mesh points around  $x^k$ . If the **poll** step also fails in finding a feasible improved mesh point, then  $x^k$  is said to be a *mesh local optimizer*. The mesh is then refined and  $x^{k+1}$  is set to  $x^k$ . If either the **search** or **poll** step succeeds in finding an improved mesh point, then the mesh size parameter is increased, and the next iterate is the improved point on the mesh.

Each iteration has two possible outcomes, leading to different parameter update rules. If a new incumbent solution is found (i.e.,  $x^{k+1} \neq x^k$  with  $f(x^{k+1}) < f(x^k)$ ), then the mesh size parameter  $\delta^{k+1}$  is increased. If no new incumbent solution is found (i.e.,  $x^{k+1} = x^k$ ), then  $\delta^{k+1}$  is decreased.

Unlike the **CS** algorithm, **GPS** allows for a general coarsening and refining of the mesh by introducing a mesh size adjustment parameter  $\tau$ . This adds no mathematical difficulty, and it can be very useful in practice. Due to the simple coarsening and refining rules, the mesh size parameter will always satisfy

$$(7.1) \quad \delta^{k+1} = (\tau)^{r^k} \delta^0$$

for some positive or negative integer  $r^k \in \mathbb{Z}$  that depends on the iteration number  $k$ . Typical default parameters are  $\delta^0 = 1$  and  $\tau = \frac{1}{2}$ .

Notice that when  $G$  is the identity matrix, and the matrix  $Z$  is composed of the positive and negative coordinate directions, and no **search** step is used, then the **GPS** algorithm looks remarkably like the **CS** algorithm. However, even in this case, the **GPS** algorithm differs from the **CS** algorithm in that when a **poll** step is successful, the mesh size parameter increases (whereas in the **CS** algorithm after successful **poll** steps the step length parameter remains the same). This illustrates the first difference between the **CS** and **GPS** algorithms: in the **GPS** algorithm the mesh size parameter both increases and decreases. Increasing the mesh size parameter allows the algorithm to recover from a bad initial choice of the parameter  $\delta^0$  (Exercise 7.2).

The second change from the **CS** to the **GPS** algorithm is the addition of the **search** step. While the **search** step is limited to examining finitely many points on the mesh, it is entirely open beyond that. The **search** step allows for the **GPS** algorithm to include embedded heuristics without breaking

the convergence analysis. For example, after every failed poll step we could apply a few iterations of a genetic algorithm to try and break free from local minimisers. This flexibility improves the practical efficiency of the method and does not affect the theoretical convergence analysis as the latter depends solely on the fact that the poll step explores in a systematic way near the incumbent solution.

The final difference between the CS and GPS algorithms is that in the GPS algorithm the set of directions  $\mathbb{D}$  is more flexible. That is, the matrix  $G$  does not need to be the identity and  $Z$  does not need to be composed of the positive and negative coordinate directions. In GPS, the set  $\mathbb{D}$  is fixed, but the set  $\mathbb{D}^k \subseteq \mathbb{D}$  can change from iteration to iteration. Thus, there is great deal of flexibility in how  $\mathbb{D}$  and  $\mathbb{D}^k$  are constructed.

**EXAMPLE 7.1.** Figure 7.1 illustrates examples of positive spanning sets  $\mathbb{D}^k$  formed by the columns of  $D = GZ$ , along with the meshes that they generate when  $\delta^k = \frac{1}{2}$ . The mesh points are at the intersections of the lines and the arrows represent the directions in  $\mathbb{D}$ .

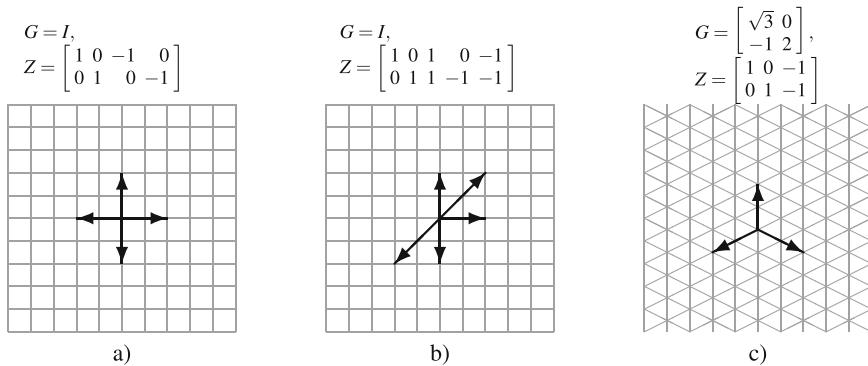


FIGURE 7.1. Examples of search sets and meshes in  $\mathbb{R}^2$  with  $\delta^k = \frac{1}{2}$  obtained by the directions  $D = GZ$ ; the mesh points are at the intersections of the lines and the arrows represent the directions in  $\mathbb{D}$ ; the poll set a) is that of the CS algorithm

The set of poll directions  $\mathbb{D}^k$  can be created by selecting any subset of  $\mathbb{D}$  that forms a positive spanning set. Mesh a) and mesh b) are identical, even though they are constructed from different sets of directions. Mesh a) is created using the positive and negative coordinate directions

The positive spanning sets used to generate a) and c) are both positive bases in  $\mathbb{R}^2$ ; however, the positive spanning set used to generate b) is not a positive basis (as it is not positive linearly independent). Nonetheless, this is a valid selection for  $D$  in the GPS algorithm. In fact, this set has some advantages, as the positive spanning set  $\mathbb{D}^k$  selected in the poll step can differ from one iteration to the next.

**EXAMPLE 7.2.** As another example, suppose that the  $2 \times 8$  matrix  $D$  is formed through

$$\mathbb{D} = \{[d_1, d_2]^\top \neq [0, 0]^\top : d_1, d_2 \in \{-1, 0, 1\}\} \subset \mathbb{R}^2.$$

The mesh is identical to that used by the CS algorithm, but with GPS there are a total of 14 distinct positive bases that can be constructed from  $\mathbb{D}$  (Exercise 7.1). In Chapter 8, we will see how to make the number of positive bases grow arbitrarily large with that simple mesh.

---

## 7.2. Opportunistic Strategy, the search Step, and Starting Points Selection

As with the CS algorithm, GPS can apply *opportunistic* strategies during the **search** and **poll** steps (see Section 3.6). That is, as soon as a trial point  $t \in S^k \cup P^k$  with  $f(t) < f(x^k)$  is found, then the algorithm can consider the step a success and terminate the iteration.

Also similar to the CS algorithm, the GPS algorithm is a local optimization method, i.e., the GPS algorithm starts at an initial point and seeks a highly accurate local minimiser. As such, a good starting point can greatly improve the effectiveness of the GPS algorithm. Section 3.4 discusses how to select a starting point for the CS algorithm, and all of the information in that section can be applied directly to the GPS algorithm as well. For example, Latin hypercube sampling (LHS, Algorithm 3.4) remains a viable method to select a starting point.

However, unlike CS, the GPS algorithm has an additional tool to break free of local minimisers and seek the global minimiser: the **search** step. The **search** step has an extremely high importance in practice. It is essential to locate promising regions in the space of variables, it allows the algorithm to escape local solutions, and it can accelerate convergence to a refined point. The **search** step also allows practitioners to apply any favourite search heuristics that they wish during the algorithm, which can be of great value in some circumstances. It is often the case that the practitioner has an in-depth knowledge of the optimization problem at hand, and that he has already devised another efficient methodology to search for a good solution.

Two simple subroutines that could be run during the **search** step are:

- i. A line search.
- ii. A few iterations of an heuristic method.

Line searches are at the heart of many optimization methods (we will revisit them in Chapter 10). In DFO/BBO, a rudimentary line search could be performed during the **search** step as follows: *if the previous iteration was successful (regardless of if the success was due to the search or poll step), try a point further along the direction that lead to the success.* For example,

consider a GPS algorithm where the mesh size parameter doubles at successful iterations (i.e.,  $\tau = \frac{1}{2}$ ). One could define the search set in Step 2 of GPS to be the singleton

$$S^k = \{x^k + 2(x^k - x^{k-1})\}.$$

This trial point belongs to the mesh  $M^k$ , because  $x^k \in M^k$  thus  $x^k = x^{k-1} + \delta^{k-1}Dz$  for some vector  $z \in \mathbb{N}^n$ , and therefore, the trial point in  $S^k$  satisfies

$$x^k + 2(x^k - x^{k-1}) = x_k + 2\delta^{k-1}Dz = x_k + \delta^k Dz \in M_k.$$

We observe in practice that consecutive successful iterations with this line search strategy can improve significantly the objective function value with a very small number of function evaluations. Consecutive successes with the line search may make significant changes to the incumbent solution.

Integrating the **search** step of the GPS algorithm requires slightly more than a cut-and-paste of a heuristic method. In particular, it is of great importance that **all points examined in the search step must lie on the mesh**. Without this property, Lemma 7.3 (below) can fail, and the entire convergence proof will fall apart. In the case of the GA method, forcing the **search** points to lie on the mesh can be done by using properly selected encoding (Section 4.3). For other heuristics such as Tabu, VNS, or Simulated Annealing, **search** points may need to be moved onto the mesh (for example through a projection operator). Chapter 13 discusses ways to exploit models and surrogates within the **search** step.

On a final note, one common strategy is to only use a heuristic **search** step at iteration  $k = 0$ . The rationale is that sampling is expensive in terms of the number of blackbox evaluations, and should be used from the start to identify a promising region in the space of variables. In this case, it should be thought of as finding an initial point.

### 7.3. The Mesh

The mesh is a discrete set of points from which the GPS algorithm selects candidate trial points. The mesh is conceptual, and is never constructed. Its coarseness is parameterised by the mesh size parameter  $\delta^k > 0$ . As the algorithm unfolds, the values of the sequence  $\delta^k$  will vary. The goal of each iteration is to obtain a mesh point whose objective function value is lower than the incumbent value  $f(x^k)$ . With this in mind, it should be no surprise that all iterates of the GPS algorithm must lie on the mesh.

**LEMMA 7.1 (GPS Iterates Lie on Mesh).**

Let  $G \in \mathbb{R}^{n \times n}$  be invertible,  $Z \in \mathbb{Z}^{n \times p}$  be such that the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^n$ , and  $D = GZ$ . Let  $M^k$  be the mesh centred at  $x^k$  with mesh size parameter  $\delta^k$ ,

$$M^k = \{x^k + \delta^k Dy : y \in \mathbb{N}^p\}.$$

Then  $x^{k+1} \in M^k$ .

**PROOF.** If the next iterate is found through a `search` step then the result holds by definition. If the next iterate is found through a `poll` step, then  $x^{k+1} = x^k + \delta^k d$  for some  $d \in \mathbb{D}^k$ , that is  $x^{k+1} = x^k + \delta^k D e_i$  for some coordinate direction  $e_i \in \mathbb{N}^p$ . Finally, if the iteration fails, then  $x^{k+1} = x^k$  trivially belongs to  $M^k$ .  $\square$

A fundamental property of the mesh, implied by the structure of  $D$ , is that the distance between pairs of distinct mesh points is bounded below by a fixed multiple of  $\delta^k$ . Hence,  $\delta^k$  represents the coarseness of the mesh.

**LEMMA 7.2 (Minimal Distance Between Mesh Points).**

Let  $G \in \mathbb{R}^{n \times n}$  be invertible,  $Z \in \mathbb{Z}^{n \times p}$  be such that the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^n$ , and  $D = GZ$ . Let  $M^k$  be the mesh centred at  $x^k$  with mesh size parameter  $\delta^k > 0$ . Then,

$$\min_{u \neq v \in M^k} \|u - v\| \geq \frac{\delta^k}{\|G^{-1}\|}.$$

**PROOF.** Let  $u$  and  $v$  be distinct points of the mesh  $M^k$ . By definition, there exist  $y_u \in \mathbb{N}^p$  and  $y_v \in \mathbb{N}^p$  with  $y_u \neq y_v$ , such that  $u = x^k + \delta^k D y_u$  and  $v = x^k + \delta^k D y_v$ . Therefore,

$$\|u - v\| = \delta^k \|D(y_u - y_v)\| = \delta^k \|GZ(y_u - y_v)\|.$$

For any invertible matrix  $G$  and vector  $d$ , the inequality  $\|G^{-1}\| \cdot \|d\| \geq \|G^{-1}d\|$  follows from the definition of induced matrix norms (see Section 2.1). This yields

$$\delta^k \|GZ(y_u - y_v)\| \geq \delta^k \frac{\|G^{-1}GZ(y_u - y_v)\|}{\|G^{-1}\|} = \delta^k \frac{\|Z(y_u - y_v)\|}{\|G^{-1}\|}.$$

Since  $Z(y_u - y_v)$  is a nonzero integer vector, it must have norm at least 1. Therefore,

$$\|u - v\| \geq \delta^k \frac{\|Z(y_u - y_v)\|}{\|G^{-1}\|} \geq \frac{\delta^k}{\|G^{-1}\|}.$$

$\square$

Perhaps surprising is the fact that if  $D$  is not constructed using the product of an invertible matrix  $G$  and an integer matrix  $Z$ , then the previous result could be false, as illustrated by the following example in  $\mathbb{R}^1$ .

**EXAMPLE 7.3.** Let  $G = 1$  and  $Z = [-1 \ \pi]$ . Then  $G$  is an invertible matrix and the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^1$ . However,  $Z$  is not an integer matrix. Let  $D = GZ$ . We will show that the mesh constructed using  $D$  centred at  $x^0 = -3 + \pi$  with coarseness  $\delta = 1$  contains infinitely many points in the interval  $[0, x^0]$ .

Consider the sequence recursively defined as  $x^{k+1} = \lceil 1/x^k \rceil x^k - 1$  for  $k = 0, 1, 2, \dots$ , where  $\lceil \cdot \rceil$  is the ceiling operator (which rounds up to the nearest integer). Observe that the sequence is composed of irrational numbers,

because  $x^0$  is irrational. By recursion, if  $x^k > 0$ , then

$$0 = \frac{1}{x^k} x^k - 1 < \left\lceil \frac{1}{x^k} \right\rceil x^k - 1 = x^{k+1},$$

(the strict inequality is due to the fact that 0 is rational and  $x^{k+1}$  is irrational) and moreover,

$$x^{k+1} = \left\lceil \frac{1}{x^k} \right\rceil x^k - 1 < \left( \frac{1}{x^k} + 1 \right) x^k - 1 = x^k.$$

It follows that  $x^0 > x^1 > \dots > x^k > \dots > 0$ . Thus we have an infinite sequence in the interval  $[0, x^0]$ .

Next we show that this sequence lies on the mesh  $\{x^0 + Dy : y \in \mathbb{N}^p\}$ . To do this, we must show that  $x^k$  is a positive integer combination of  $-1$  and  $\pi$ , i.e.,  $x^k = -a^k + b^k\pi$  for some  $a^k$  and  $b^k$  in  $\mathbb{N}$ . The result is true for  $k = 0$  with  $a^0 = 3$  and  $b^0 = 1$ . By recursion, suppose that the result is true for a given  $k \geq 0$ . Then setting  $a^{k+1} = \lceil 1/x^k \rceil a^k + 1 \in \mathbb{N}$  and  $b^{k+1} = \lceil 1/x^k \rceil b^k \in \mathbb{N}$  yields

$$-a^{k+1} + b^{k+1}\pi = -\left(\left\lceil \frac{1}{x^k} \right\rceil a^k + 1\right) + \left\lceil \frac{1}{x^k} \right\rceil b^k\pi = \left\lceil \frac{1}{x^k} \right\rceil x^k - 1 = x^{k+1}.$$

This shows that it would be possible that the sequence of iterates produced by applying GPS on minimising the smooth function  $f(x) = x$  converges to the origins with only successful search steps. The origin is not a stationary point.

In conclusion, this example shows that if the matrix  $D$  (formed by the columns of  $\mathbb{D}$ ) cannot be constructed as the product of a nonsingular matrix  $G$  with an integer positive spanning set  $Z$ , then the distance between distinct points of the resulting mesh may be arbitrarily small.

#### 7.4. Convergence

Similar to the convergence analysis of CS presented in Section 3.5, the analysis for GPS is essentially divided into two steps. First, we show that the mesh size becomes infinitely fine. Second, we show that limit points satisfy some first order optimality conditions.

Lemma 7.2 showed that the minimal distance over all pairs of distinct mesh points is bounded below by the mesh size parameter  $\delta^k$  times a scalar. We now show that there is a subsequence of unsuccessful iterates for which the mesh size parameter goes to zero. This is separated into two parts. First, the mesh size parameters generated by the GPS algorithm must be bounded above, and second that a subsequence of mesh size parameters must converge to 0. In both cases, we assume that the objective function  $f$  has bounded level sets.

**LEMMA 7.3 (Upper Bound on the Mesh Coarseness).**

Let  $\{\delta_k\}$  be the sequence of mesh size parameters produced by applying the GPS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Then, there exists a positive integer  $\hat{z}$  such that  $\delta^k \leq \delta^0(\tau)^{\hat{z}}$  for every integer  $k \geq 0$ .

**PROOF.** All iterates produced by GPS belong to the level set  $\mathcal{L}(f(x^0)) = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ . Since  $\mathcal{L}(f(x^0))$  is bounded, there exists  $\gamma > 0$  such that  $\mathcal{L}(f(x^0)) \subseteq B_\gamma(x^0)$ . Now, if the mesh size parameter is large enough so that  $\delta^k > 2\gamma\|G^{-1}\|$ , then Lemma 7.2 with  $v = x^k$  ensures that any trial point  $u \in M^k$  different from  $x^k$  would lie outside of  $\mathcal{L}(f(x^0))$ .

It follows that any iteration whose mesh size parameter exceeds  $2\gamma\|G^{-1}\|$  is unsuccessful. If iteration  $k - 1$  is successful, and if  $\delta^{k-1} \leq 2\gamma\|G^{-1}\| < \delta^k$ , then iteration  $k$  will necessarily be unsuccessful and

$$\delta^{k+1} = \delta^{k-1} \leq 2\gamma\|G^{-1}\| < \delta^k = \delta^{k-1}\tau^{-1} \leq 2\gamma\|G^{-1}\|\tau^{-1}.$$

Thus, every  $\delta^k$  is bounded above by  $2\gamma\|G^{-1}\|\tau^{-1}$  and the result follows by selecting the integer  $\hat{z}$  so that it satisfies  $\delta^0(\tau)^{\hat{z}} \geq 2\gamma\|G^{-1}\|\tau^{-1}$ .  $\square$

The next theorem states that there is a subsequence of mesh size parameters that gets arbitrarily close to zero. Since the mesh is refined only at unsuccessful iterations, this implies that there is a subsequence of iterates for which  $x^{k_i}$  is a mesh local optimizer and  $\delta^{k_i}$  goes to 0.

**THEOREM 7.4 (The Mesh Gets Infinitely Fine).**

Under the conditions of Lemma 7.3, the sequence of mesh size parameters satisfies  $\liminf_{k \rightarrow +\infty} \delta^k = 0$ .

**PROOF.** Suppose by way of contradiction that there exists an integer  $\hat{z}$  such that  $0 < \delta^0(\tau)^{\hat{z}} \leq \delta^k$  for all  $k \geq 0$ . By Lemma 7.3 and the GPS parameter update rules, we have that for any  $k \geq 0$ ,  $\delta^k = \delta^0(\tau)^{r^k}$  where  $r^k$  takes its value from the integers of the finite set  $\{\hat{z}, \hat{z} + 1, \dots, \check{z}\}$ , for some  $\check{z} \in \mathbb{N}$ .

Since  $x^{k+1} \in M^k$  (Lemma 7.1), we know that  $x^{k+1} = x^k + \delta^k Dz^k$  for some  $z^k \in \mathbb{N}^p$ . Using Lemma 7.3 by substituting  $\delta^k = \delta^0(\tau)^{r^k}$  it follows that for any integer  $N \geq 1$ ,

$$\begin{aligned} x^N &= x^0 + \sum_{k=0}^{N-1} \delta^k Dz^k = x^0 + \delta^0 D \sum_{k=0}^{N-1} (\tau)^{r^k} z^k \\ &= x^0 + \frac{(p)^{\hat{z}}}{(q)^{\hat{z}}} \delta^0 D \sum_{k=0}^{N-1} (p)^{r^k - \hat{z}} (q)^{\hat{z} - r^k} z^k \end{aligned}$$

where  $p$  and  $q$  are relatively prime integers satisfying  $\tau = \frac{p}{q}$ . Since for any  $k$  the term  $(p)^{r^k - \hat{z}} (q)^{\hat{z} - r^k} z^k$  appearing in this last sum is an integer, it follows

that all iterates lie on the translated integer lattice generated by  $x^0$  and the columns of the matrix  $\frac{(p)^z}{(q)^z} \delta^0 D$ .

Since all iterates belong to the level set  $\mathcal{L}(f(x^0))$ , and  $f$  has bounded level sets, we know all iterates belong to a compact set. The intersection of a translated integer lattice and a compact set is finite. Thus, it follows that there are only finitely many different iterates, and one of them must be visited infinitely many times. But the iteration can never return to an earlier iterate once it has left that point because of the requirement that the objective function value decreases in a successful iteration. Therefore increase rule ( $\delta^{k+1} = \tau^{-1} \delta^k$ ) is only applied finitely many times, so the decrease rule ( $\delta^{k+1} = \tau \delta^k$ ) is applied infinitely many times. This contradicts the hypothesis that  $\delta^0(\tau)^z$  is a lower bound for the mesh size parameter.  $\square$

The rationality of  $\tau$  is used in the previous proof to generate a translated integer lattice. It turns out that this is necessary to complete the proof. Indeed, Exercise 7.12 shows that if  $\tau$  is irrational, then  $\delta^k$  can be bounded away from 0.

Now that we have a subsequence of the mesh size parameters converging to 0, we formally define *mesh local optimizers*.

**DEFINITION 7.2 (Mesh Local Optimizer).**

*The point  $x^k$  is called a mesh local optimizer for the GPS algorithm if and only if both the search step and poll step fail at iteration  $k$ .*

Theorem 7.4 shows that an infinite subsequence of mesh local optimizers exist, because the mesh size parameter decreases only at failed iterations. We now turn our attention to how the function behaves on that subsequence.

**PROPOSITION 7.5.**

*Let  $\{x^k\}$  be the sequence of iterates produced by applying the GPS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Then,*

- i. *there exists at least one limit point of the iteration sequence  $\{x^k\}$ , and*
- ii. *if  $f \in \mathcal{C}^0$  then every limit point has the same objective function value.*

**PROOF.** i. As  $\mathcal{L}(f(x^0))$  is bounded, its closure  $\text{cl}(\mathcal{L}(f(x^0)))$  (recall Definition 2.6) is compact. The algorithm generates an infinite sequence in this set, therefore must have a convergent subsequence.

- ii. Suppose  $x^{k_i} \rightarrow \bar{x}$  ( $i \in K_1$ ) and  $x^{k_j} \rightarrow \tilde{x}$  ( $j \in K_2$ ). Since  $f \in \mathcal{C}^0$ , we have  $\lim_{i \rightarrow \infty} f(x^{k_i}) = f(\bar{x})$  and  $\lim_{j \rightarrow \infty} f(x^{k_j}) = f(\tilde{x})$ . Since  $\{f(x^k)\}$  is a nonincreasing sequence, and  $\{f(x^{k_j})\}$  is an infinite subsequence of  $\{f(x^k)\}$ , we must have  $\{f(x^k)\}$  bounded below by  $f(\tilde{x})$ . Thus  $f(\bar{x}) \geq f(\tilde{x})$ . Applying the same logic to  $\{f(x^{k_i})\}$ , we have  $f(\tilde{x}) \geq f(\bar{x})$ . So every limit point has the same objective function value.  $\square$

Theorem 7.4 gives us an infinite sequence of mesh local optimizers, and Proposition 7.5 shows that they all converge to the same objective function value (provided  $f \in \mathcal{C}^0$ ). Our next goal is to show first order optimality holds for the accumulation points of any convergent subsequence. The following definition will be useful in defining the limit points we wish to study.

**DEFINITION 7.3** (Refining Subsequences and Refined Points).

*Let  $\{x^k\}_{k \in K}$  be a convergent subsequence of mesh local optimizers.*

*The subsequence is said to be a refining subsequence if and only if  $\lim_{k \in K} \delta^k = 0$ . The limit of a refining subsequence is called its corresponding refined point.*

As shown in Exercise 3.8, the iteration sequence as a whole need not converge, and there may even be infinitely many limit points. The next result guarantees the existence of a refining subsequence and its refined point.

**THEOREM 7.6** (Refined Points Exist).

*If  $\{x^k\}$  be the sequence of iterates produced by applying the GPS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets, then there exist a refining subsequence  $\{x^k\}_{k \in K}$  and a refined point  $\hat{x}$ .*

**PROOF.** Let  $\{x^k\}$  be the sequence of iterates produced by applying the GPS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Define  $U$  as the set of indices of unsuccessful iterations. Theorem 7.4 ensures that  $U$  contains an infinite number of elements, because the mesh size parameter is decreased only at unsuccessful iterations. Therefore, one can extract an infinite subset  $V \subseteq U$  such that  $\lim_{k \in V} \delta^k = 0$ . All iterates  $x^k$  with  $k \in V$  belong to a bounded level set, and therefore we can extract an infinite subset  $K \subseteq V$  such that  $\{x^k\}_{k \in K}$  converges. Let  $\hat{x}$  denote its limit.

It follows that  $\{x^k\}_{k \in K}$  is a convergent subsequence of mesh local optimizers (because  $K \subseteq U$ ) that satisfies  $\lim_{k \in K} \delta^k = 0$  (because  $K \subseteq V$ ). Therefore  $\{x^k\}_{k \in K}$  is a refining subsequence, and  $\hat{x}$  is a refined point.  $\square$

To prove more, we will need to assume more. However, instead of assuming continuous differentiability as we did for the analysis of CS, we will begin with the weaker assumption of Lipschitz continuity, and explore the generalised directional derivatives at any refined point.

**THEOREM 7.7 (Convergence of GPS).**

Let  $\{x^k\}$  be the sequence of iterates produced by applying the GPS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Let  $\{x^k\}_{k \in K}$  be a refining sequence and  $\hat{x}$  the corresponding refined point.

- i. If  $f$  is Lipschitz continuous, then for any direction  $d \in \mathbb{D}$  such that  $f$  was evaluated infinitely many times in the subsequence  $K$  during a poll step we have  $f^\circ(\hat{x}; d) \geq 0$ .
- ii. If  $f$  is Lipschitz continuous and regular at  $\hat{x}$ , then for any direction  $d \in \mathbb{D}$  such that  $f$  was evaluated infinitely many times in the subsequence  $K$  during a poll step we have  $f'(\hat{x}; d) \geq 0$ .
- iii. If  $f \in \mathcal{C}^1$ , then

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

**PROOF.** i. For a locally Lipschitz function  $f$ , we have by the definition of generalised directional derivative and of  $\limsup$  that:

$$f^\circ(\hat{x}; d) = \limsup_{y \rightarrow \hat{x}, t \searrow 0} \frac{f(y + td) - f(y)}{t} \geq \limsup_{k \in K} \frac{f(x^k + \delta^k d) - f(x^k)}{\delta^k}.$$

Since, for each  $k \in K$ ,  $x^k$  is a refining point, we have  $\delta^{k+1} = \tau \delta^k$ . This occurs if and only if  $f(x^k + \delta^k d) \geq f(x^k)$  for all  $d \in \mathbb{D}^k$ . Hence, for any  $d \in \mathbb{D}$  where  $f$  was evaluated infinitely many times during a poll step in the subsequence  $K$ , we have

$$\limsup_{k \in K} \frac{f(x^k + \delta^k d) - f(x^k)}{\delta^k} \geq 0,$$

and the conclusion follows.

- ii. If, in addition,  $f$  is regular at  $\hat{x}$ , then the result follows immediately from the fact that  $f^\circ(\hat{x}; d) = f'(\hat{x}; d)$  for all  $d \in \mathbb{R}^n$  (Definition 6.5).
- iii. If  $f \in \mathcal{C}^1$ , then  $f^\circ(\hat{x}; d) = f'(\hat{x}; d) = \nabla f(\hat{x})^\top d$  for all  $d \in \mathbb{R}^n$ . Since the number of ways to select a positive bases from  $\mathbb{D}$  is finite, there must exist at least one positive basis that is selected an infinite number of times within the subsequence  $K$ . Let  $\mathbb{D}'$  be that positive basis, so there exists an infinite subsequence  $K' \subseteq K$  such that  $\mathbb{D}^k = \mathbb{D}'$  for all  $k \in K'$ . Applying part (i) to every vector in  $\mathbb{D}'$  we have  $\nabla f(\hat{x})^\top d \geq 0$  for all  $d \in \mathbb{D}'$ . Theorem 6.5 ensures that  $\nabla f(\hat{x}) = 0$ .  $\square$

Theorem 7.7 part i. is the key to our analysis. The fact that its proof follows so directly from Clarke's definition of the generalised directional derivative is because unsuccessful polling at mesh local optimizers belonging to convergent refining sequences provide exactly the nonnegative difference quotients that generalised directional derivatives need. We believe that this illustrates an intimate relationship between generalised directional derivatives and the GPS algorithm.

Note that while Theorem 7.7 provides a broader convergence analysis than Theorem 3.4 did, we are not claiming that condition  $f \in \mathcal{C}^1$  implies that the GPS algorithm *always* finds a minimiser. The GPS algorithm can suffer from the same flaw as the CS algorithm; namely, if the wrong search directions are employed, then it is possible to converge to a noncritical point. Simple convex counterexamples come from starting at just the wrong point and choosing just the right ill-suited directions. This can be seen in Example 3.3, or by considering  $f(x) = |x_1| + |x_2|$  on  $\mathbb{R}^2$ , starting with  $x^0 = [1, 0]^\top$ ,  $\mathbb{D} = \{[1, 0]^\top, [-1, 1]^\top, [-1, -1]^\top\}$ , and using no `search` step. In the later case, the initial point  $x^0$  is a mesh local optimizer for every  $\delta > 0$ , and so the iteration never moves from  $x^0$ .

In summary, if the level sets of  $f$  are bounded, then the GPS algorithm guarantees the following hierarchy of convergence behaviour.

- i. With no additional assumptions, there must be at least one refining subsequence and refined point  $\hat{x}$ .
- ii. If  $f \in \mathcal{C}^0$ , then every refined point has the same objective function value.
- iii. If  $f$  is Lipschitz near a refined point  $\hat{x}$ , then the generalised directional derivatives satisfy  $f^\circ(\hat{x}; d) \geq 0$  for all directions  $d$  in the positive basis  $\mathbb{D}'$ .
- iv. If  $f$  is Lipschitz near a refined point  $\hat{x}$  and regular at  $\hat{x}$ , then the directional derivatives satisfy  $f'(\hat{x}; d) \geq 0$  for all directions  $d$  in the positive basis  $\mathbb{D}'$ .
- v. If  $f \in \mathcal{C}^1$ , then  $\nabla f(\hat{x}) = 0$  for any refined point  $\hat{x}$ .

Note that stronger differentiability assumptions do not necessarily guarantee stronger convergence results. The following example, in  $\mathbb{R}^2$ , shows that even if  $f \in \mathcal{C}^1$ , then the mesh size parameter does not necessarily converge to zero, although the limit inferior does. There can be limit points of the GPS algorithm sequence that are not critical points. This example does not contradict Theorem 7.7, as all limit points still have the same objective function value, and the noncritical limit points are not refined points.

**EXAMPLE 7.4.** Consider the function  $f$  defined as follows,

$$\begin{aligned} f(a, b) = & b^2((a-1)^2 + 2) - 2(\max\{b-a, 0\})^2 \\ & + 8(\min\{a, 0\})^2 + (a-1)(\max\{a-\frac{1}{2}, 0\})^2. \end{aligned}$$

It can be shown that  $f$  is continuously differentiable everywhere on  $\mathbb{R}^2$  (Exercise 7.10). We apply the GPS algorithm using an empty search step, opportunistic poll step, and parameters

$$x^0 = [a^0, b^0]^\top = [0, 1]^\top, \quad \delta^0 = \frac{1}{4}, \quad \tau = \frac{1}{2},$$

$$G = I(\text{the identity matrix}), \quad D = Z = \begin{bmatrix} 1 & 0 & -1 & 2 & -1 \\ 0 & 1 & -2 & -2 & 0 \end{bmatrix}.$$

The positive basis  $\mathbb{D}^k$  used at iteration  $k$  varies at each iteration as follows (where  $d_j$  is the  $j^{th}$  column of  $D$  and where  $x^k = [a^k, b^k]^\top$ )

$$\mathbb{D}^k = \begin{cases} \{d_4, d_2, d_3\} & \text{if } b^k = 4\delta^k \\ \{d_5, d_2, d_4\} & \text{if } b^k \neq 4\delta^k \text{ and } a^k = 1 \\ \{d_1, d_2, d_3\} & \text{otherwise.} \end{cases}$$

Figure 7.2 shows some level sets of  $f$  and plots the first 21 trial points generated by the algorithm. Iterations 22, 23, ..., 27 have the incumbent solution unchanged, and iteration 28 moves the incumbent solution to  $x^{28} = [1/16, 1/16]^\top$ .

The key structure of this example is that for any fixed value of  $b > 0$ , if  $a \geq b$  and  $a = 2^{-i}$  with  $i \in \mathbb{N}$ , then

$$f(a, b) > f(2a, b) > f(4a, b) > f(8a, b) > \dots > f(2^{-1}, b) > f(1, b).$$

Furthermore, for any  $b > 0$  we have  $f(1, b) = 2b^2 > b^2 = f(0, b)$ .

As a result, starting from the incumbent solution  $[a, b]^\top$ , the algorithm generates a series of successful iterations going through  $[2a, b]^\top, [4a, b]^\top$  up to  $[1, b]^\top$ , and then it uses the direction  $d_5$  with a mesh size parameter equal to 1 to move to  $[0, b]^\top$ . Then, a series of unsuccessful poll steps reduces the mesh size parameter to the value  $b/4$  and polling in the direction  $d_4$  is successful. This generates the new incumbent solution  $[a/2, a/2]^\top$ . The entire process restarts from that point.

The resulting sequence of iterates possesses an infinite number of accumulation points, namely all points of the set

$$\{[0, 0]^\top\} \cup \{[2^{-\ell}, 0]^\top : \ell = 0, 1, \dots\}.$$

Each of these accumulation points share the same objective function value, but  $[0, 0]^\top$  is the only refined point. The gradient at all these points is zero, except for  $[1, 0]^\top$  where  $\nabla f(1, 0) = [1/4, 0]^\top$ . This does not contradict Theorem 7.7, as the subsequence converging to  $[1, 0]^\top$  is not a refining subsequence.

This example highlights that Theorem 7.4 cannot be strengthened, as it is possible that the limit superior of the mesh size parameters is strictly positive. Indeed, in this example the limit superior of the sequence of mesh size parameters is 2.

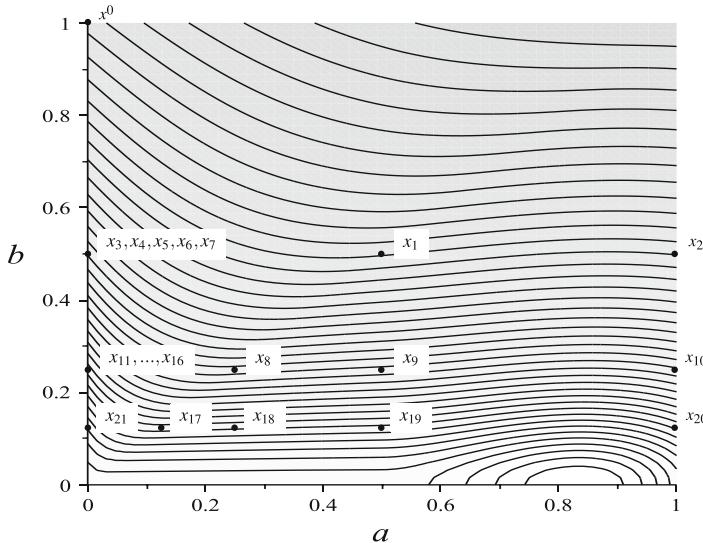


FIGURE 7.2. The GPS algorithm can create noncritical accumulation points

### 7.5. Numerical Experiments with the Rheology Problem

We end this chapter by applying the GPS algorithm to the rheology parameter fit optimization problem from Section 1.3.3. Since we are working with direct search methods, we focus on the nonsmooth rheology parameter fitting problem. As we see, the GPS algorithm provides a significant improvement over the CS algorithm.

**EXAMPLE 7.5.** Let us revisit the nonsmooth rheology parameter fit optimization problem from Section 1.3.3, as it was reformulated in Example 3.1:

$$\min_{x \in \mathbb{R}^3} \{\hat{f}(x) : x \in [\ell, u]\}.$$

This example was analyzed with the GS and CS algorithms in Examples 3.1 and 3.5.

We apply the GPS algorithm using the coordinate directions, which results in a variant of the CS algorithm where the mesh size parameter doubles on successful iterations. We label this **GPS COORDINATE**. We also apply the GPS algorithm using

$$D = \frac{1}{3} \begin{bmatrix} -2\sqrt{2} & \sqrt{2} & \sqrt{2} & 0 & 2\sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 \\ 0 & -\sqrt{6} & \sqrt{6} & 0 & 0 & \sqrt{6} & -\sqrt{6} & 0 \\ 1 & 1 & 1 & -3 & -1 & -1 & -1 & 3 \end{bmatrix}$$

and label it **GPS 2n + 2**.

We ran these two GPS variants from the same 7 starting points from Example 3.5, whose function values are also listed that example. All runs are performed with the opportunistic strategy that reorders the polling directions

when the iteration is successful with the same additional budgets of 125 and 375 function evaluations. Table 7.1 presents the results and includes the results of the CS algorithm with ordered polling for ease of comparison.

TABLE 7.1. Best objective function value  $\hat{f}$  on the non-smooth rheology problem, generated by CS, GPS COORDINATE and GPS  $2n + 2$  from different initial points

| Initial Point        | CS                   |                      | GPS COORDINATE       |                      | GPS $2n + 2$         |                      |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
|                      | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls |
| $x_{\text{GS}}^0$    | 299.883              | 299.883              | 299.884              | 299.883              | 238.833              | 238.172              |
| $x_{\text{LHS}_1}^0$ | 200.930              | 199.190              | 170.844              | 158.981              | 95.642               | 54.057               |
| $x_{\text{LHS}_2}^0$ | 220.553              | 220.546              | 214.802              | 214.749              | 195.907              | 189.44               |
| $x_{\text{LHS}_3}^0$ | 215.069              | 215.069              | 215.877              | 211.743              | 163.388              | 163.284              |
| $x_{\text{LHS}_4}^0$ | 136.042              | 136.040              | 136.943              | 136.040              | 163.053              | 161.897              |
| $x_{\text{LHS}_5}^0$ | 46.132               | 46.129               | 54.729               | 46.129               | 179.967              | 177.964              |
| $x_{\text{LHS}_6}^0$ | 170.080              | 170.080              | 170.090              | 170.080              | 163.953              | 163.896              |

The analysis of the table reveals some trends, many similar to the observations following the analysis of Table 3.3 devoted to CS. Once again, the GS strategy to generate the initial point is worst than every LHS initialisation. Also, the improvement to the solution obtained by increasing the number of additional function evaluations from 125 to 375 is often marginal. The most important improvements are for the GPS  $2n + 1$  variant, in particular from  $x_{\text{LHS}_1}^0$ .

With a budget of 375 additional function evaluations, GPS COORDINATE dominates CS. The only algorithmic difference is that the mesh size parameter doubles at successful iterations rather than staying the same.

There is great variability from one starting point to another, but GPS  $2n + 2$  outperforms both other instances for five out of the seven starting points.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 7.1.** Consider the  $2 \times 8$  matrix  $D$  is formed through

$$\mathbb{D} = \{[d_1, d_2]^\top \neq [0, 0]^\top : d_1, d_2 \in \{-1, 0, 1\}\} \subset \mathbb{R}^2.$$

The

- a) Show that  $D$  can be created as the product of an invertible matrix and an integer matrix whose columns are a positive spanning set for  $\mathbb{R}^2$ .

- b) Prove that there are a total of 14 distinct positive bases of  $\mathbb{R}^2$  that can be constructed from  $\mathbb{D}$  (8 minimal and 6 maximal).

**EXERCISE 7.2.** Let  $x^* \in \mathbb{R}_+$  be the minimiser of the convex function of a single variable  $f(x) = \max \left\{ 100 + \frac{1}{1+|x|}, \frac{|x|}{100} \right\}$ .

- a) What is the value of  $x^*$ ?
- b) Starting from  $x^0 = 0$  with  $\delta^0 = 1$ , how many function evaluations will CS require before it generates an incumbent solution  $x^k$  that satisfies  $|x^k - x^*| < 1$ ?
- c) Starting from  $x^0 = 0$  with  $\delta^0 = 1$ ,  $\tau = \frac{1}{2}$  and  $D = [I - I]$ , how many function evaluations will GPS (with an empty search set) require before it generates an incumbent solution  $x^k$  that satisfies  $|x^k - x^*| < 1$ ?

**EXERCISE 7.3.** Consider the matrix

$$D = \begin{bmatrix} 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \end{bmatrix}.$$

- a) Sketch  $\mathbb{D}$ , the set of column vectors of  $D$ .
- b) How many distinct positive spanning sets of  $\mathbb{R}^2$  can be constructed from  $\mathbb{D}$ ?
- c) How many distinct positive bases of  $\mathbb{R}^2$  can be constructed from  $\mathbb{D}$ ?

**EXERCISE 7.4.** Consider a GPS algorithm in which  $\mathbb{D}^k = \{d \in \mathbb{Z}^n : \|d\|_\infty = 1\}$  and in which the poll step is complete, i.e., the objective function is evaluated at all points of  $P^k$  at iteration  $k$ .

- a) Show that  $\mathbb{D}^k$  is a positive spanning set of  $\mathbb{R}^n$ .
- b) Give a reason why one should not implement such an algorithm.

**EXERCISE 7.5.** Suppose that

$$M = \{[x, y, z]^\top : x \in \mathbb{Z}, y \in \mathbb{Z}, z \in \mathbb{Z}\} \subseteq \mathbb{R}^3,$$

is the mesh generated by  $D = GZ$  centred at  $x^0 = [0, 0, 0]^\top$  with coarseness  $\delta^0 = 1$ . Suppose  $G$  is the  $3 \times 3$  identity matrix  $I$ , and the columns of  $Z \in \mathbb{Z}^{3 \times p}$  form a positive spanning set of  $\mathbb{R}^3$ .

- a) Explain why for the above to hold, we must have  $Z \notin \mathbb{Z}^{3 \times 3}$ . (i.e., show  $p > 3$ .)
- b) Give examples where the above holds and  $Z \in \mathbb{Z}^{3 \times p}$  with  $p = 4, 5, \dots, 8$ .
- c) Explain why there is no upper bound on the dimensions of  $Z$ .

**EXERCISE 7.6.** + Consider a GPS algorithm in which the poll set is formed using a minimal positive basis  $\mathbb{D}^k$  of  $\mathbb{R}^n$ , such that the sum of its  $n + 1$  elements is the null vector, and in which the mesh size adjustment parameter is set to  $\frac{1}{2}$ . Following an unsuccessful iteration, the next search step orders the  $n + 1$  unsuccessful poll points and labels them  $y^0, y^1, \dots, y^n$  and defines  $x^r, x^e, x^{oc}$ , and  $x^{ic}$  as in the NM algorithm from Chapter 5.

- a) Propose some values of the NM parameters  $\delta^e, \delta^{ic}$ , and  $\delta^{oc}$  ensuring that following an unsuccessful iteration, the points  $x^r, x^e, x^{oc}$ , and  $x^{ic}$  lie on the mesh  $M^k$ .
- b) Describe a GPS search step that mimics the logic of the NM algorithm.

**EXERCISE 7.7.** Suppose that

$$M = \mathbb{Z}^n,$$

is the mesh generated by  $D = GZ$  centred at  $x^0 = 0$  with coarseness  $\delta^0 = 1$ . Suppose  $G$  is the identity matrix  $I$ , and the columns of  $Z \in \mathbb{Z}^{n \times p}$  form a positive spanning set of  $\mathbb{R}^n$ . Prove  $p > n$ .

**EXERCISE 7.8.** Let  $\mathbb{D} = \{d_1, d_2, \dots, d_p\} \subset \mathbb{Z}^p$  be a positive spanning set for  $\mathbb{R}^n$ , and let  $G \in \mathbb{R}^{n \times n}$  be an invertible matrix. For  $j = 1, 2, \dots, p$  define  $d'_j = Gd_j$ , and define  $\mathbb{D}' = \{d'_1, d'_2, \dots, d'_p\}$ . Show that  $\mathbb{D}'$  is a positive spanning set for  $\mathbb{R}^n$ .

**EXERCISE 7.9.**  $\square$  Let  $G = 1$ ,  $Z = [-1 \ U]$ ,  $U \in \mathbb{R}$ ,  $D = GZ$ , and consider the mesh generated by  $D$  centred at  $x^0 = 0$  with coarseness  $\delta^0 = 1$ . We can sketch a portion of this mesh by selecting a range parameter  $R > 0$ , and computing all values  $p$  such that

$$p_i = -a + bU, \text{ for some } a \in \mathbb{Z}, |a| \leq R, b \in \mathbb{Z}, |b| \leq R.$$

- a) Sketch the resulting portion of the mesh when  $Z = [-1 \ \pi]$  using  $R = 10$ ,  $R = 100$ , and  $R = 1000$ . Repeat and restrict the sketch to the values  $p \in [-1, 1]$ .
- b) Sketch the resulting portion of the mesh when  $Z = [-1 \ 3.1]$  using  $R = 10$ ,  $R = 100$ , and  $R = 1000$ . Repeat and restrict the sketch to the values  $p \in [-1, 1]$ .
- c) Sketch the resulting portion of the mesh when  $Z = [-1 \ 22/7]$  using  $R = 10$ ,  $R = 100$ , and  $R = 1000$ . Repeat and restrict the sketch to the values  $p \in [-1, 1]$ .
- d) Theoretically what should the results of part b) and c) look like? If your results do not match the theory, explain why.

**EXERCISE 7.10.**  $\square$  Consider Example 7.4.

- a) Analytically prove that

$$\begin{aligned} f(a, b) &= b^2((a-1)^2 + 2) - 2(\max\{b-a, 0\})^2 \\ &\quad + 8(\min\{a, 0\})^2 + (a-1)(\max\{a-1/2, 0\})^2. \end{aligned}$$

is continuously differentiable on  $\mathbb{R}^2$ : i.e.,  $f \in \mathcal{C}^1$ .

- b) Numerically confirm the first 8 iterates  $(x^0, x^1, \dots, x^8)$ .

**EXERCISE 7.11.** + Let  $\ell \in \mathcal{C}^1$  and  $u \in \mathcal{C}^1$ . Let  $f$  be a function and  $\hat{x}$  be fixed. Suppose there exists  $\epsilon > 0$  such that

$$\ell(\hat{x}) = f(\hat{x}) = u(\hat{x}) \quad \text{and} \quad \ell(x) \leq f(x) \leq u(x) \quad \forall x \in B_\epsilon(\hat{x}).$$

Prove that, if  $\nabla \ell(\hat{x}) = \nabla u(\hat{x})$ , then all directional derivatives of  $f$  exist at  $\hat{x}$  and  $f'(\hat{x}; d) = u'(\hat{x}; d)$  for all  $d \in \mathbb{R}^n$ .

**EXERCISE 7.12.** + We apply an algorithm similar to GPS with the coordinate directions but with an irrational parameter  $\tau = \frac{1}{\sqrt{3}}$  and with  $\delta^0 = 1$  on the discontinuous objective function

$$f(x) = \begin{cases} 3^{-j} & \text{if } x = \sqrt{3}j - [\sqrt{3}(j-1)] \quad \text{for some } j \in \mathbb{N} \\ 2 \times 3^{-(j+1)} & \text{if } x = \sqrt{3}j - [\sqrt{3}j] \quad \text{for some } j \in \mathbb{N} \\ 1 & \text{otherwise.} \end{cases}$$

When the iteration number  $k$  is a multiple of 4, the search set is the singleton  $S^k = \{x^k - \lfloor x^k \rfloor \delta^k\}$ . Otherwise the search set is empty.

- a) Show that  $f(x) = 1$  when  $x \in (-\infty, 0) \cup [1, \sqrt{3}) \cup [1 + \sqrt{3}, +\infty)$ .
- b) Show that if  $x^k = \sqrt{3}j - [\sqrt{3}(j-1)]$  for some  $j \in \mathbb{N}$  and  $\delta^k = 1$  then

the search step is a success and  $x^{k+1} = \sqrt{3}j - [\sqrt{3}j]$ ,  
 $\delta^{k+1} = \sqrt{3}$ ;

the poll step is a success and  $x^{k+2} = \sqrt{3}(j+1) - [\sqrt{3}j]$ ,  
 $\delta^{k+2} = 3$ ;

the iteration is unsuccessful and  $x^{k+3} = \sqrt{3}(j+1) - [\sqrt{3}j]$ ,  
 $\delta^{k+3} = \sqrt{3}$ ;

the iteration is unsuccessful and  $x^{k+4} = \sqrt{3}(j+1) - [\sqrt{3}j]$ ,  
 $\delta^{k+4} = 1$ .

- c) Use induction to show that for any  $j \in \mathbb{N}$ , the sequence of mesh size parameters generated by GPS from the starting point  $x^0 = 1$  satisfies  $\delta^{4j} = 1$ ,  $\delta^{4j+1} = \delta^{4j+3} = \sqrt{3}$  and  $\delta^{4j+2} = 3$ .
- d) Conclude that if the parameter  $\tau$  is irrational, then it is possible that the mesh size parameter is bounded below by some strictly positive constant.

**EXERCISE 7.13.** + Provide details proving the paragraph from Example 7.4,

Starting from the incumbent solution  $[a, b]^\top$ , the algorithm will generate a series of successful iterations going through  $[2a, b]^\top, [4a, b]^\top$  up to  $[1, b]^\top$ , and then it will use the direction  $d_5$  to move to  $[0, b]^\top$ . Then, a series of unsuccessful poll steps will reduce the mesh size parameter to the value  $b/4$  and polling in the direction  $d_4$  will be successful. This will lead to a new incumbent solution of  $[a/2, a/2]^\top$ . The entire process will restart from that point.



**Chapter 7 Project: Apply GPS to the Nonsmooth Rheology Problem.** Implement the GPS algorithm.

- a) Apply the GPS algorithm on the rheology problem (from Section 1.3.3) using the formulation from Example 7.5. Explain how (and why) your results differ from those in Table 3.3.

- b) Apply the GPS algorithm on the rheology problem using

$$D = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 \\ 0 & 1 & 0 & -1 & 1 & -1 & 1 & -1 \end{bmatrix},$$

and  $\mathbb{D}^k$  generated by the following rules

$$\begin{aligned} \mathbb{D}^0 &= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \\ \mathbb{D}^{k+1} &= \mathbb{D}^k, && \text{if iteration } k \text{ succeeds} \\ \mathbb{D}^{k+1} &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbb{D}^k, && \text{if iteration } k \text{ fails.} \end{aligned}$$

Explain how your results compare to the approaches in Table 3.3.

# *Mesh Adaptive Direct Search*

Chapter 3 proposed the coordinate search (CS) algorithm for unconstrained optimization. The algorithm worked based on local exploration around the incumbent solution in the positive and negative orthogonal coordinate directions. In Chapter 7, this algorithm was expanded to allow a broader use of search directions, resulting in the generalised pattern search (GPS) algorithm. However, the convergence analysis of the GPS algorithm falls a little short of what we would like. First, the main result states that if the objective function  $f$  is locally Lipschitz, then the generalised directional derivatives are nonnegative for only a *finite set* of directions. Second, and more importantly, the analysis only examined unconstrained optimization problems. In practice, there are very few problems in which the variables are free to take any values.

In this chapter we present an advancement on the GPS algorithm, the Mesh Adaptive Direct Search (MADS) algorithm. We will show convergence for both unconstrained and constrained optimization. The MADS algorithm will also address the limitation in the GPS algorithm of only proving that a finite list of directional derivatives is nonnegative. This shortcoming is

because the initialisation stage of the GPS algorithm requires a finite positive spanning set from which all future positive bases will be selected, effectively forcing the algorithm to only consider a finite list of potential polling directions. The MADS algorithm will generalise the GPS algorithm by allowing an infinite set of polling directions, while improving the convergence results.

### 8.1. The MADS Algorithm

In order to present the MADS algorithm, it is necessary to recall the definition of a mesh, and introduce the definition of the *frame*. In the GPS algorithm this second definition was unnecessary, as the *frame size parameter* and the *mesh size parameter* were both represented by the same single parameter  $\delta^k > 0$ . The main change from the GPS algorithm to the MADS algorithm originates from this separation. Let us recall the definition of a mesh and formalise the definition of the mesh size parameter.

**DEFINITION 8.1 (Mesh and Mesh Size Parameter).**

Let  $G \in \mathbb{R}^{n \times n}$  be invertible and  $Z \in \mathbb{Z}^{n \times p}$  be such that the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^n$ . Define  $D = GZ$ . The mesh of coarseness  $\delta^k > 0$  generated by  $D$ , centred at the incumbent solution  $x^k \in \mathbb{R}^n$ , is defined by

$$M^k := \{x^k + \delta^k Dy : y \in \mathbb{N}^p\} \subset \mathbb{R}^n,$$

where  $\delta^k$  is called the mesh size parameter.

In the GPS algorithm, the mesh size parameter is used to control the poll step, by creating the *poll set* (renamed with a subscript)

$$P_{\text{GPS}}^k = \{x^k + \delta^k d : d \in \mathbb{D}_{\text{GPS}}^k\},$$

where  $\mathbb{D}_{\text{GPS}}^k$  is a positive basis selected from  $\mathbb{D}$  the columns of the matrix  $D$ . The basic idea of the MADS algorithm is to create a new parameter, the frame size parameter  $\Delta^k$ , that replaces  $\delta^k$  in the poll step. The frame size parameter  $\Delta^k$  defines the frame in which polling is done. At every iteration, these two parameters will satisfy  $0 < \delta^k \leq \Delta^k$ .

**DEFINITION 8.2 (Frame and Frame Size Parameter).**

Let  $G \in \mathbb{R}^{n \times n}$  be invertible and  $Z \in \mathbb{Z}^{n \times p}$  be such that the columns of  $Z$  form a positive spanning set for  $\mathbb{R}^n$ . Define  $D = GZ$ . Select a mesh size parameter  $\delta^k > 0$  and let  $\Delta^k$  be such that  $\delta^k \leq \Delta^k$ . The frame of extent  $\Delta^k$  generated by  $D$ , centred at the incumbent solution  $x^k \in \mathbb{R}^n$ , is defined by

$$F^k := \{x \in M^k : \|x - x^k\|_\infty \leq \Delta^k b\}$$

with  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$  and where  $\Delta^k$  is called the frame size parameter.

Notice that  $P_{\text{GPS}}^k \subseteq F^k \subseteq M^k$  (Exercise 8.1). In the MADS algorithm, the poll set is constructed by selecting mesh points from inside the frame

$F^k$ , and is not necessarily forced to equal  $P_{\text{GPS}}^k$ . By having the mesh size parameter decrease more rapidly than the frame size parameter, this allows for asymptotically dense selections of poll directions (formalised in Section 8.4).

The idea of a frame is fairly technical and can be difficult to grasp at first glance. Therefore, before proceeding to a statement of the MADS algorithm, let us study a quick example in  $\mathbb{R}^2$ .

**EXAMPLE 8.1.** Let

$$D = \begin{bmatrix} 1 & 0 & 1 & 1 & -1 & 0 & -1 & -1 \\ 0 & 1 & 1 & -1 & 0 & -1 & -1 & 1 \end{bmatrix}.$$

Figure 8.1 illustrates some meshes, frames, and potential poll sets in  $\mathbb{R}^2$  for four values of  $\delta^k$  and  $\Delta^k$ . The mesh points are located at the intersection of the thin grey lines, while the frames are the mesh points inside of the region delimited by the thick black boxes. The three poll points are  $p^1, p^2, p^3$ .

In the first example (Figure 8.1 top left), both  $\delta^k = 1$  and  $\Delta^k = 1$ . In this case, both the MADS and GPS algorithms select the poll set from the eight neighbouring mesh points.

In the other examples, we have  $\delta^k < \Delta^k$ . The GPS algorithm would still be forced to select the poll set from the eight neighbouring mesh points. However, the MADS algorithm can use any point in the frame, other than  $x^k$ , to create polling directions ( $x^k$  is excluded as it would result in a polling direction of 0). In the top-right figure,  $\delta^k = \frac{1}{4}$  and  $\Delta^k = \frac{1}{2}$  and the MADS algorithm may select a poll set from 24 points in the frame (remember to exclude  $x^k$ ). In the bottom left of Figure 8.1, the MADS algorithm would have 80 points to select from, and in the bottom right this number increases to 288.

More generally, if for some positive integer  $p$ , the frame size parameter equals  $2^{-p}$  and the mesh size parameter equals  $4^{-p}$ , then there would be  $(2^{p+1} + 1)^2$  points in the frame, creating  $(2^{p+1} + 1)^2 - 1$  potential polling directions (Exercise 8.2).

Allowing the mesh size parameter to decrease at a faster rate than the frame size parameter opens up the possibility to use a richer set of polling directions. The MADS algorithm presented below uses a fairly simple parameter update strategy:  $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ . Other update strategies are also possible.

The MADS algorithm targets constrained optimization problems of the form

$$\min_x \{f(x) : x \in \Omega\}.$$

Of course,  $\Omega = \mathbb{R}^n$  is perfectly reasonable. Constraints are treated by a rather unsophisticated strategy, called the *extreme barrier* which involves the extended valued barrier function.

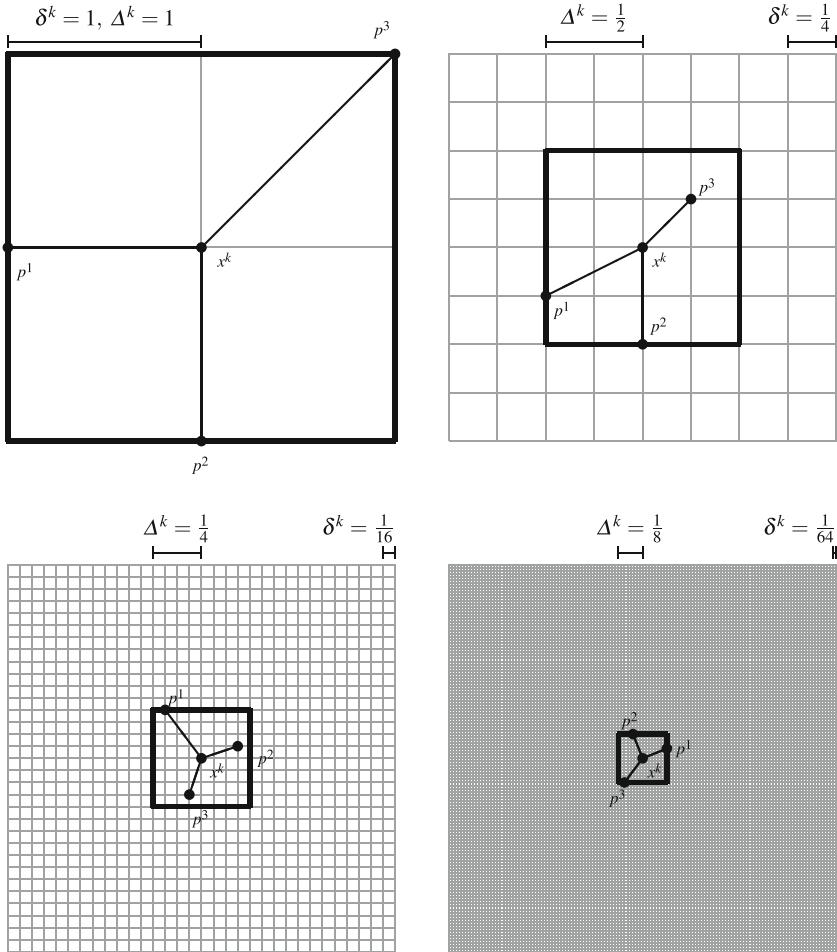


FIGURE 8.1. Example of meshes and frames in  $\mathbb{R}^2$  for different values of  $\delta^k$  and  $\Delta^k$

**DEFINITION 8.3 (Extreme Barrier Function).**

Given an objective function  $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  and a constraint set  $\Omega \subseteq \mathbb{R}^n$ , the extreme barrier function  $f_\Omega : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  is defined as

$$f_\Omega(x) := \begin{cases} f(x) & \text{if } x \in \Omega, \\ \infty & \text{if } x \notin \Omega. \end{cases}$$

The optimization is conducted by using the value of the barrier function  $f_\Omega$  rather than  $f$ , and therefore any feasible point will be *de facto* preferable

to infeasible ones. Thus, MADS treats the unconstrained optimization problem  $\min_x \{f_\Omega(x) : x \in \mathbb{R}^n\}$ . More subtle ways of handling constraints that exploit the functions defining  $\Omega$  are presented in Chapter 12.

**ALGORITHM 8.1.** Mesh adaptive direct search (MADS)

---

Given  $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  and starting point  $x^0 \in \Omega$  and barrier function  $f_\Omega(x)$

0. Initialisation

|  |                                |
|--|--------------------------------|
| $\Delta^0 \in (0, \infty)$               | initial frame size parameter   |
| $D = GZ$                                 | positive spanning matrix       |
| $\tau \in (0, 1)$ , with $\tau$ rational | mesh size adjustment parameter |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance             |
| $k \leftarrow 0$                         | iteration counter              |

1. Parameter Update

| set the mesh size parameter to  $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$

2. Search

| if  $f_\Omega(t) < f_\Omega(x^k)$  for some  $t$  in a finite subset  $S^k$  of  $M^k$   
|     set  $x^{k+1} \leftarrow t$  and  $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$  and go to 4  
| otherwise go to 3

3. Poll

| select a positive spanning set  $\mathbb{D}_\Delta^k$  such that  
|      $P^k = \{x^k + \delta^k d : d \in \mathbb{D}_\Delta^k\}$   
| is a subset of the frame  $F^k$  of extent  $\Delta^k$   
| if  $f_\Omega(t) < f_\Omega(x^k)$  for some  $t \in P^k$   
|     set  $x^{k+1} \leftarrow t$  and  $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$   
| otherwise  
|     set  $x^{k+1} \leftarrow x^k$  and  $\Delta^{k+1} \leftarrow \tau \Delta^k$

4. Termination

| if  $\Delta^{k+1} \geq \epsilon_{\text{stop}}$   
|     increment  $k \leftarrow k + 1$  and go to 1  
| otherwise stop

---

The MADS algorithm is similar to the GPS algorithm in many ways. For example, the search step is restricted to the mesh, and the mesh size parameter is reduced after unsuccessful iterations and increased after successful iterations. The major difference lies in the poll step. In the GPS algorithm the poll set is created by taking directions from the set  $\mathbb{D}$  and then scaling them by length  $\delta^k$ . In the MADS algorithm, the poll set is created in a slightly different way, as the directions no longer need to be taken from  $\mathbb{D}$ . Instead, the algorithm is permitted to select any direction such that  $x^k + \delta^k d$  lies inside the frame. In the GPS algorithm the magnitude of  $\delta^k d$ ,  $d \in \mathbb{D}_{\text{GPS}}^k$ , is trivially bounded by  $\delta^k \|d\| \leq \delta^k b$  where  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$ . Alternatively, in

the MADS algorithm, we have  $\delta^k d \leq \Delta^k b$  for all  $d \in \mathbb{D}_\Delta^k$ . Since  $\Delta^k > \delta^k$  whenever  $\delta^k < 1$ , this allows for a greater flexibility in how the poll set is constructed.

As a side effect of the new structure, unlike the GPS algorithm, the MADS algorithm does not require a large complicated matrix  $D$  in order to create flexibility in the poll steps.

**EXAMPLE 8.2.** Let

$$D = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}.$$

The meshes and frames created for this matrix are identical to those of Example 8.1. However, if the GPS algorithm were used, then the top left of Figure 8.1 is no longer an acceptable poll set, as it is not a subset of  $\mathbb{D}$ . This shows that the MADS algorithm can produce complex structures even with minimal positive spanning sets.

---

## 8.2. Opportunistic Strategy, the search Step, and Starting Points Selection

Like the GPS algorithm, the MADS algorithm can apply the opportunistic strategy during the search and poll steps. It is a local optimization method that includes a search step to break free of local minimisers. Thus, all of the comments of Section 7.2 apply directly to the MADS algorithm.

## 8.3. Convergence Analysis of MADS

The convergence analysis of the MADS algorithm uses the definitions and results presented in both Sections 6.4 and 6.5.

Like the GPS algorithm, the convergence analysis for the MADS algorithm will use the standard assumption that the level sets of  $f$  are bounded. Since the MADS algorithm allows for a constraint set  $\Omega$ , and since the extreme barrier approach is used, we shall also use the standard assumption that a feasible initial point is given, i.e.,  $x^0 \in \Omega$ . This assumption is discarded in Chapter 12 where another strategy to handle constraints is presented.

Recall that the GPS algorithm's convergence analysis was divided into two parts. First, we showed that  $\liminf_{k \rightarrow \infty} \delta^k = 0$ , then we showed that some generalised directional derivatives were nonnegative at any *refined point*. We take a similar approach now. First we show that both  $\liminf_{k \rightarrow \infty} \delta^k = 0$  and  $\liminf_{k \rightarrow \infty} \Delta^k = 0$ .

**THEOREM 8.1 (Mesh Gets Infinitely Fine).**

Let  $\{\delta_k\}$  and  $\{\Delta^k\}$  be the sequence of mesh size and frame size parameters produced by applying the MADS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Then, the sequences satisfy

$$\liminf_{k \rightarrow \infty} \Delta^k = \liminf_{k \rightarrow \infty} \delta^k = 0.$$

Moreover,  $\{\delta^k\}_{k \in K}$  is a subsequence (for some subset of indices  $K$ ) with  $\lim_{k \in K} \delta^k = 0$  if and only if  $\{\delta^k\}_{k \in K}$  is a subsequence with  $\lim_{k \in K} \Delta^k = 0$ .

**PROOF.** First note that Lemmas 7.1 and 7.3 still hold for the MADS algorithm. Indeed, Lemma 7.1 holds trivially, and Lemma 7.3 holds as it only relied on the fact that all evaluated points must lie on the mesh. With this in mind, the proof of Theorem 7.4 holds, replacing GPS by MADS as appropriate. This leads to  $\liminf_{k \rightarrow \infty} \delta^k = 0$ . The remainder of the Theorem follows from  $0 < \delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ .  $\square$

The second part of the GPS algorithm's convergence analysis studied the generalised directional derivative at *refined points* of the algorithm. For the MADS algorithm we shall add the definition of a *refining direction* to that of a *refining subsequence* and a *refined point*. This new definition is necessary because the directions are not anymore confined to a finite set.

**DEFINITION 8.4 (Refining Subsequences, Points and Directions).**

A convergent subsequence of mesh local optimizers  $\{x^k\}_{k \in K}$  (for some subset of indices  $K$ ) is said to be a refining subsequence if and only if  $\lim_{k \in K} \delta^k = 0$ . The limit  $\hat{x}$  of  $\{x^k\}_{k \in K}$  is called a refined point.

Given a refining subsequence  $\{x^k\}_{k \in K}$  and its corresponding refined point  $\hat{x}$ , a direction  $d$  is said to be a refining direction if and only if there exists a infinite subset  $L \subseteq K$  with poll directions  $d^k \in \mathbb{D}_\Delta^k$  such that  $x^k + \delta^k d^k \in \Omega$  and  $\lim_{k \in L} \frac{d^k}{\|d^k\|} = \frac{d}{\|d\|}$ .

The existence of at least one refining subsequence along with a refining direction follows similar logic to Theorem 7.6. Begin with the facts that  $\liminf_{k \rightarrow \infty} \delta^k = 0$  and  $f$  has bounded level sets to obtain a refining subsequence and refining point. Next note that, by Theorem 8.1, if  $\{x^k\}_{k \in K}$  is a refining subsequence, then  $\lim_{k \in K} \Delta^k = 0$  (in addition to  $\lim_{k \in K} \delta^k = 0$ ). The existence a refining direction  $d$  now follows from the fact that the closed ball is compact. In addition, if  $d$  is an hypertangent direction to  $\Omega$  at  $x$ , then Definition 6.7 guarantees that  $x^k + \delta^k d^k \in \Omega$  for all sufficiently large  $k \in L$  (Exercise 8.6).

The following Lemma gives a lower bound on the generalised directional derivative.

**LEMMA 8.2 (Lower Bound on  $f^\circ$ ).**

Suppose  $f : \Omega \mapsto \mathbb{R}^n$  is Lipschitz near  $\hat{x} \in \Omega \subseteq \mathbb{R}^n$ , and  $d \in T_\Omega^H(\hat{x})$  is an hypertangent direction, then

$$f^\circ(\hat{x}; d) \geq \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{f(x + tv) - f(x)}{t}.$$

**PROOF.** Denoting the Lipschitz constant of  $f$  by  $M$  yields

$$\begin{aligned} & \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{f(x + tv) - f(x)}{t} \\ &= \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{f(x + tv) - f(x + td) + f(x + td) - f(x)}{t} \\ &\leq \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{f(x + td) - f(x)}{t} \\ &\quad + \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{|f(x + tv) - f(x + td)|}{t} \\ &\leq f^\circ(x; d) + \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} \frac{M\|tv - td\|}{t} \\ &= f^\circ(\hat{x}; d) + \limsup_{x \rightarrow \hat{x}, v \rightarrow d, t \searrow 0} M\|v - d\| = f^\circ(\hat{x}; d). \end{aligned}$$

The above inequality used the fact that  $x + tv$  and  $x + td$  belong to  $\Omega$  when  $t > 0$  is small, which follows from the hypothesis that  $d$  is an hypertangent direction.  $\square$

We now present our basic result on refining directions from which our hierarchy of results is derived.

**THEOREM 8.3 (Convergence of MADS).**

Let  $f$  be Lipschitz near a refined point  $\hat{x} \in \Omega$ , and  $d \in T_\Omega^H(\hat{x})$  be a refining direction for  $\hat{x}$ . Then the generalised directional derivative of  $f$  at  $\hat{x}$  in the hypertangent direction  $d$  is nonnegative, i.e.,  $f^\circ(\hat{x}; d) \geq 0$ .

**PROOF.** Let  $\{x^k\}_{k \in K}$  be a refining subsequence converging to the refined point  $\hat{x}$  and let  $d \in T_\Omega^H(\hat{x})$  be a refining direction. Therefore, there exists an infinite subsequence  $L$  of the set of indices  $K$  of unsuccessful iterations, with poll directions  $d^k \in \mathbb{D}_\Delta^k$  such that  $\lim_{k \in L} \frac{d^k}{\|d^k\|} = \frac{d}{\|d\|}$ . Furthermore, Definition 6.7 of an hypertangent vector guarantees that  $x^k + \delta^k d^k \in \Omega$  for all sufficiently large  $k \in L$ .

Now, applying Lemma 8.2 using sequences  $x^k \rightarrow \hat{x}$ ,  $d^k/\|d^k\| \rightarrow d/\|d\|$ , and  $\delta^k \|d^k\| \searrow 0$  (as  $\delta^k \|d^k\| \leq \Delta^k b \searrow 0$ ), we find that

$$\begin{aligned} f^\circ\left(\hat{x}; \frac{d}{\|d\|}\right) &\geq \limsup_{x \rightarrow \hat{x}, v \rightarrow d/\|d\|, t \searrow 0} \frac{f(x + tv) - f(x)}{t} \\ &\geq \limsup_{k \in L} \frac{f\left(x^k + \delta^k \|d^k\| \frac{d^k}{\|d^k\|}\right) - f(x^k)}{\delta^k \|d^k\|} \geq 0, \end{aligned}$$

because  $x^k$  is a refining subsequence, so  $x^k + \delta^k d^k \in \Omega$  resulted in an unsuccessful poll step, i.e.,  $f(x^k + \delta^k d^k) = f_\Omega(x^k + \delta^k d^k) \geq f_\Omega(x^k) = f(x^k)$  for all  $k \in L$ .  $\square$

Theorem 8.3 makes an important and subtle advancement from the convergence theory of the GPS algorithm. Specifically, Theorem 8.3 shows convergence for a constrained optimization problem. Interestingly, the algorithm deals with the constraint set by creating a new (unconstrained) problem of minimising  $f_\Omega$ . However, the convergence results in Theorem 8.3 are linked to the local smoothness of  $f$  and not  $f_\Omega$ . This is important, as obviously  $f_\Omega$  is discontinuous on the boundary of  $\Omega$ , so not Lipschitz continuous there.

Reexamining Proposition 7.5, it is easy to see that the results also hold for MADS (Exercise 8.4). As such, we now easily prove the following hierarchy of convergence behaviour for the MADS algorithm.

- i. With no additional assumptions, there must be at least one refining subsequence and refined point  $\hat{x}$ .
- ii. If  $f \in \mathcal{C}^0$ , then every refined point has the same objective function value.
- iii. If  $f$  is Lipschitz near a refined point  $\hat{x}$ , then the generalised directional derivatives satisfy  $f^\circ(\hat{x}; d) \geq 0$  for any refining direction  $d \in T_\Omega^H(\hat{x})$ .
- iv. If  $f$  is Lipschitz near a refined point  $\hat{x}$  and regular at  $\hat{x}$ , then the directional derivatives satisfy  $f'(\hat{x}; d) \geq 0$  for any refining direction  $d \in T_\Omega^H(\hat{x})$ .
- v. If  $f \in \mathcal{C}^1$  and  $\Omega$  is convex near a refined point  $\hat{x}$ , then  $v^\top \nabla f(\hat{x}) \geq 0$  for any direction  $v \in \text{conv}(V)$  where  $V = \{d \in T_\Omega^H(\hat{x}) : d \text{ is a refining direction}\}$ .

Notice that in the absence of constraints, or when the refined point belongs to the interior of  $\Omega$ , then the results ii., iii., and iv. above generalise to all (generalised) directional derivatives are nonnegative, and to the gradient equals zero.

## 8.4. Dense Sets of Polling Directions

So far, the convergence results of the MADS algorithm seem very similar to those of the GPS algorithm. Indeed, both Theorem 7.7 and Theorem 8.3 basically say that any direction that is sampled in the limit of a refining sequence must have a nonnegative directional derivative. The key difference is for the GPS algorithm the directions come from a finite set; while, for the MADS algorithm the directions are generated from an ever expanding set

of options. This crucial distinction comes from the fact that the mesh size parameter  $\delta^k$  goes to zero more rapidly than the poll size parameter  $\Delta^k$ .

To see the full power of this, we demonstrate that the MADS algorithm can actually produce an asymptotically dense set of poll directions. Before proceeding, let us generalise the definition of a dense subset (Definition 3.1) and formalise the definition.

**DEFINITION 8.5 (Asymptotically Dense).**

The set  $V \subseteq \mathbb{R}^n$  is said to be asymptotically dense if the normalised set  $\{v/\|v\| : v \in V\}$  is dense on the unit sphere  $S = \{w \in \mathbb{R}^n : \|w\| = 1\}$ .

In order to demonstrate that the MADS algorithm can actually produce an asymptotically dense set of poll directions, we shall make use of the *Householder matrix*.

**DEFINITION 8.6 (Householder Matrix).**

Let  $v \in \mathbb{R}^n$  be a normalised vector. The Householder matrix  $H$  associated with  $v$  is

$$H := I - 2v v^\top \in \mathbb{R}^{n \times n},$$

where  $I$  is the  $n \times n$  identity matrix.

**EXAMPLE 8.3.** Consider the normalised vector  $v = [3/5, -4/5]^\top \in \mathbb{R}^2$ . To find the associated Householder matrix set

$$\begin{aligned} H &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - 2 \begin{bmatrix} 3/5 \\ -4/5 \end{bmatrix} [3/5 \ -4/5] \\ &= \frac{1}{25} \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix} - \frac{2}{25} \begin{bmatrix} 9 & -12 \\ -12 & 16 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} 7 & 24 \\ 24 & -7 \end{bmatrix}. \end{aligned}$$

Under the  $\ell_2$  norm, the columns of  $H$  satisfy

$$\left\| \frac{1}{25} \begin{bmatrix} 7 \\ 24 \end{bmatrix} \right\| = 1, \quad \left\| \frac{1}{25} \begin{bmatrix} 24 \\ -7 \end{bmatrix} \right\| = 1, \quad \text{and} \quad [7 \ 24] \begin{bmatrix} 24 \\ -7 \end{bmatrix} = 0.$$

That is, the columns of  $H$  form an orthonormal basis<sup>1</sup> of  $\mathbb{R}^2$ .

---

The final comment in Example 8.3 is not a coincidence. In fact, given any nonzero vector in  $\mathbb{R}^n$ , the columns of the associated Householder matrix will form an orthonormal basis of  $\mathbb{R}^n$  (Exercise 8.11). Combining this with Theorem 6.4, we have the following result.

---

<sup>1</sup>An orthonormal basis is composed of orthogonal and unit vectors.

**COROLLARY 8.4** (Householder Positive Basis).

Let  $v$  be a nonzero vector in  $\mathbb{R}^n$  and  $H$  be the associated Householder matrix. Then for any  $\gamma \neq 0$  the columns of  $\gamma H[I - I]$  forms a maximal positive basis of  $\mathbb{R}^n$ .

Our strategy to show that the MADS algorithm can produce asymptotically dense poll directions is as follows.

- (1) We shall set  $\Delta^0 = 1$ ,  $\tau = \frac{1}{2}$ , and  $\mathbb{D}$  to be the set of positive and negative coordinate directions in  $\mathbb{R}^n$ .
- (2) We shall assume that we have dropped to a subsequence such that the frame size parameter  $\Delta^k \searrow 0$ , and the mesh size parameter  $\delta^k = (\Delta^k)^2$  at all iterations.
- (3) Next, we build dense sequence  $\{v^k\}_{k=1}^\infty$  in the unit sphere.
- (4) To this sequence, we apply the Householder matrix to create a sequence of polling directions that can be used within the MADS algorithm.
- (5) Finally, we show that the resulting positive bases create asymptotically dense poll directions.

The choices in Step 1 are clearly valid. The assumption of dropping to a refining sequence in Step 2 is justified by Theorem 8.1. Building a sequence  $V = \{v^k\}_{k=1}^\infty$  that is a dense in the sphere  $S \subset \mathbb{R}^n$  is achievable, and one example of how to do this is given in Exercise 8.8. Thus, we shall accept Step 3 as justified.

Therefore, our next step is to use the Householder matrix to create poll directions that can be used within the MADS algorithm. The exact construction is summarised in the next subroutine.

**ALGORITHM 8.2.** Creating the set of poll directions

Given  $v^k \in \mathbb{R}^n$  with  $\|v^k\| = 1$  and  $\Delta^k \geq \delta^k > 0$

1. Create Householder matrix
 

|   |                                    |
|---|------------------------------------|
| use $v^k$ to create its associated Householder matrix | $H^k = [h_1 \ h_2 \ \dots, \ h_n]$ |
|---|------------------------------------|
2. Create poll set
 

|   |  |
|---|--|
| define $\mathbb{B}^k = \{b_1, b_2, \dots, b_n\}$ with $b_j = \text{round} \left( \frac{\Delta^k}{\delta^k} \frac{h_j}{\ h_j\ _\infty} \right) \in \mathbb{Z}^n$ | $\left( \frac{\Delta^k}{\delta^k} \frac{h_j}{\ h_j\ _\infty} \right) \in \mathbb{Z}^n$ |
| set $\mathbb{D}_\Delta^k = \mathbb{B}^k \cup (-\mathbb{B}^k)$   |  |

Before proving Algorithm 8.2 creates the directions we desire, let us study an example.

**EXAMPLE 8.4.** Suppose that at iteration  $k$  the incumbent solution is the origin and the mesh and poll size parameters are  $\delta^k = \frac{1}{64}$  and  $\Delta^k = \frac{1}{8}$ . Suppose  $\mathbb{D}$  is the set of positive and negative coordinate directions in  $\mathbb{R}^4$ ,  $D = [I - I] \in \mathbb{R}^{4 \times 8}$ , and  $v^k = \frac{1}{\sqrt{70}}[-5, 3, 6, 0]^\top$ . The corresponding Householder matrix is

$$H^k = \frac{1}{70} \begin{bmatrix} 20 & 30 & 60 & 0 \\ 30 & 52 & -36 & 0 \\ 60 & -36 & -2 & 0 \\ 0 & 0 & 0 & 70 \end{bmatrix}.$$

The  $j^{\text{th}}$  column of the matrix  $\mathbb{B}^k$  is  $b_j = \text{round}\left(\frac{64}{8} \frac{h_j}{\|h_j\|_\infty}\right) = \text{round}\left(8 \frac{h_j}{\|h_j\|_\infty}\right)$ , so

$$\mathbb{B}^k = \text{round}\left(8 \times \begin{bmatrix} 1/3 & 15/26 & 1 & 0 \\ 1/2 & 1 & -3/5 & 0 \\ 1 & -9/13 & -1/30 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 3 & 5 & 8 & 0 \\ 4 & 8 & -5 & 0 \\ 8 & -6 & 0 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}.$$

The set  $\mathbb{D}_\Delta^k$  is therefore

$$\mathbb{D}_\Delta^k = \begin{bmatrix} 3 & 5 & 8 & 0 & -3 & -5 & -8 & 0 \\ 4 & 8 & -5 & 0 & -4 & -8 & 5 & 0 \\ 8 & -6 & 0 & 0 & -8 & 6 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & -8 \end{bmatrix}.$$

The matrix  $\mathbb{B}^k$  is invertible and therefore  $\mathbb{D}_\Delta^k$  is a maximal positive basis of  $\mathbb{R}^4$  (Theorem 6.4). We now consider the first poll point

$$x^k + \frac{1}{64}[3 \ 4 \ 8 \ 0]^\top = x^k + [3/64 \ 1/16 \ 1/8 \ 0]^\top$$

and show that it belongs to the frame  $F^k$  of extent  $\Delta^k$ . Indeed, to belong to the frame, a point must be of the form  $x^k + \delta^k d$  such that  $d = Dy$ ,  $y \in \mathbb{N}^p$ , and  $\delta^k \|d\| \leq \Delta^k b$ . In this case  $D = [I - I]$ ,  $b = 1$  and  $d = [I - I][3 \ 4 \ 8 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ , so  $x^k + \delta^k d$  belongs to the mesh ( $y \in \mathbb{N}^p$ ). Finally, notice that

$$\delta^k \|d\|_\infty = \frac{1}{64} \|[3 \ 4 \ 8 \ 0]\|_\infty = \frac{1}{8} = \Delta^k b,$$

so  $x^k + \delta^k d$  lies within the frame. Using this process, it is easy to check that all poll points belong to the frame.

The results of Example 8.4 are not a coincidence. Indeed, the output of Algorithm 8.2 will lie within the frame.

**THEOREM 8.5 (Householder Positive Basis in MADS).**

Suppose  $\Delta^0 = 1$ ,  $\tau = \frac{1}{2}$ , and  $\mathbb{D}$  is the set of positive and negative coordinate directions. Suppose  $\Delta^k < 1$  and  $\delta^k = (\Delta^k)^2$ . Given  $v^k \in \mathbb{R}^n$ , let  $\mathbb{D}_\Delta^k = \mathbb{B}^k \cup (-\mathbb{B}^k)$  be the output of Algorithm 8.2. Then,

$x^k + \delta^k d$  is in the frame  $F^k$  of extent  $\Delta^k$  for all  $d \in \mathbb{D}_\Delta^k$ .

Moreover, if  $\Delta^k$  is sufficiently small, then  $\mathbb{D}_\Delta^k$  is a positive basis of  $\mathbb{R}^n$ .

**PROOF.** In order to show that  $x^k + \delta^k d$  is in the frame, we must show that  $d = [I - I]y$  for some  $y \in \mathbb{N}^{2n}$ , and that  $\delta^k \|d\|_\infty \leq \Delta^k b$  where  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\} = 1$ .

The first of these is trivial as every  $b_j$  belongs to  $\mathbb{Z}^n$  and  $d \in \{\pm b_j : j = 1, 2, \dots, n\}$ . The second of these follows from the fact that each component of the vector  $b_j = \text{round}\left(\frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty}\right)$  belongs to the interval  $\left[-\frac{\Delta^k}{\delta^k}, \frac{\Delta^k}{\delta^k}\right]$  whose endpoints satisfy  $\frac{\Delta^k}{\delta^k} = \frac{1}{\Delta^k} \in \mathbb{N}$ , and therefore there exists an index  $j$  such that  $\|d\|_\infty = \|b_j\|_\infty \leq \frac{\Delta^k}{\delta^k}$ . Thus  $x^k + \delta^k d$  belongs to the frame  $F^k$ .

To see  $\mathbb{D}_\Delta^k$  is a positive basis requires some technical details which we shall omit. The general argument is as follows. First note that the columns of  $H^k$  form a basis of  $\mathbb{R}^n$ . Thus, the vectors  $\hat{h}_j = \frac{\Delta^k}{\delta^k} \frac{h_j}{\|h_j\|_\infty}$  also form a basis for  $\mathbb{R}^n$ . After some technical details it can be shown that if  $\Delta^k$  is small, then the errors resulting from the round operator are also small, and the columns of  $\mathbb{B}^k$  forms a basis of  $\mathbb{R}^n$ . This implies  $\mathbb{B}^k$  is invertible. Thus, applying Theorem 6.4 yields that  $\mathbb{D}_\Delta^k$  is a positive basis of  $\mathbb{R}^n$ .  $\square$

Theorem 8.5 justifies Step 4 above. To justify the fifth and final step, we show that given a dense set of directions in the unit sphere  $S$ , the columns of the associated Householder matrices are asymptotically dense.

**THEOREM 8.6 (Asymptotically Dense Set of Directions).**

Let  $v^k \in \mathbb{R}^n$  be the direction used to construct the  $n \times n$  Householder matrix  $H^k$  at iteration  $k$ , and let  $\mathbb{H}^k$  be the set of columns of  $H^k$ . If  $\{v^k\}_{k=0}^\infty$  is dense in the unit sphere in  $\mathbb{R}^n$ , then  $\{h : h \in \mathbb{H}^k\}_{k=0}^\infty$  is asymptotically dense in  $\mathbb{R}^n$ .

As such, if  $\mathbb{D}_\Delta^k$  is constructed as in Algorithm 8.2, then the sequence of sets  $\mathbb{D}_\Delta^k$  form an asymptotically dense set of directions.

**PROOF.** Let  $d$  be a normalised direction in  $\mathbb{R}^n$ . Without any loss of generality, suppose that  $d_1 \neq 1$ . Construct  $w \in \mathbb{R}^n$  by setting  $w_1 = \sqrt{\frac{1-d_1}{2}}$  and  $w_i = \frac{-d_i}{2w_1}$  for each  $i = 2, 3, \dots, n$ . This vector belongs to the unit sphere because

$$\|w\| = w^\top w = \frac{1-d_1}{2} + \frac{\sum_{i=2}^n d_i^2}{4w_1^2} = \frac{1-d_1}{2} + \frac{1-d_1^2}{2(1-d_1)} = 1.$$

By the density assumption, there exists a subsequence  $\{v^k\}_{k \in K}$  converging to the normalised vector  $w$ . The first column of the Householder matrix  $H^k$  is  $h^k = e_1 - 2v_1^k v^k$  whose limit for  $k \in K$  is

$$e_1 - 2w_1 w = \begin{bmatrix} 1 - 2w_1^2 \\ -2w_1 w_2 \\ \vdots \\ -2w_1 w_n \end{bmatrix} = \begin{bmatrix} 1 - 2\left(\frac{1-d_1}{2}\right) \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = d.$$

In other words, for any direction  $d$ , there exists a subset of columns of the Householder matrix that converges to  $d$ .  $\square$

In conclusion, if the MADS set of poll directions is constructed using Algorithm 8.2 (with asymptotically dense refining directions), and if the entire sequence of incumbent solutions converges to the refined point  $\hat{x} \in \Omega$ , then the convergence analysis ensures that  $f^\circ(\hat{x}; d) \geq 0$  for every direction  $d \in T_\Omega^H(\hat{x})$  provided that  $f$  is Lipschitz near  $\hat{x}$ . Exercise 8.14 illustrates this point by proposing an instance of MADS that generates exactly two refined points. The union of all normalised poll directions is dense in the unit sphere, but the union of all refining directions at either refined point is not dense in the sphere, and neither of the refined point are minimisers.

## 8.5. Further Experiments with the Rheology Optimization Problem

Let us study one more time the nonsmooth rheology parameter fitting optimization problem from Section 1.3.3. This example was analyzed with GS, CS, and GPS in Examples 3.1, 3.5, and 7.5.

**EXAMPLE 8.5.** We apply the MADS algorithm to the nonsmooth rheology parameter fitting optimization problem using the same seven starting points as in Example 7.5

$$\min_{x \in \mathbb{R}^3} \{\hat{f}(x) : x \in [\ell, u]\}.$$

TABLE 8.1. Best objective function value  $\hat{f}$  on the non-smooth rheology problem, generated by CS, GPS, and GPS  $2n + 2$  versus MADS from different initial points

| Initial Point        | Best of CS and GPS   |                      |                      | MADS                 |                      |                      |
|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
|                      | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 875 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 875 $\hat{f}$ -calls |
| $x_{\text{GS}}^0$    | 238.833              | 238.172              | 238.171              | 231.220              | 227.636              | 220.563              |
| $x_{\text{LHS}_1}^0$ | 95.6422              | 54.0571              | 39.7329              | 177.506              | 38.735               | 33.992               |
| $x_{\text{LHS}_2}^0$ | 195.907              | 189.438              | 189.438              | 212.744              | 206.499              | 61.500               |
| $x_{\text{LHS}_3}^0$ | 163.388              | 163.284              | 163.284              | 183.367              | 181.538              | 80.717               |
| $x_{\text{LHS}_4}^0$ | 136.042              | 136.040              | 136.040              | 108.301              | 32.912               | 32.839               |
| $x_{\text{LHS}_5}^0$ | 46.132               | 46.129               | 46.129               | 59.654               | 37.393               | 33.275               |
| $x_{\text{LHS}_6}^0$ | 163.953              | 163.896              | 163.896              | 164.331              | 41.815               | 32.868               |

Table 8.1 compares the solutions produced by the best of the three runs using CS, GPS to the solution produced by MADS after 125, 375, and 875 additional function evaluation.

The first obvious observation is that MADS does not get stuck at sub-optimal solutions as does CS and GPS. The MADS polling directions change from one iteration to another, and eventually contain a descent direction. With a larger computational budget of evaluations, MADS would move away from nonstationary points. Figure 8.2 shows level sets of  $\hat{f}$  by varying two out of the three variables. The objective function is clearly not differentiable. The central dot represents the final solution produced by the GPS algorithm with coordinate directions. One can see that with the coordinate directions, it is impossible to improve that point.

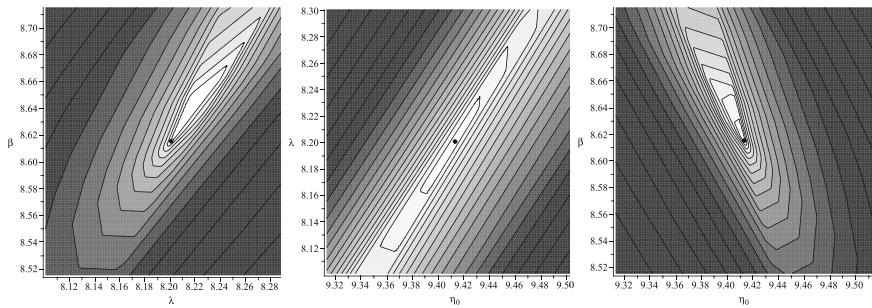


FIGURE 8.2. Level sets of the objective function value  $f$  of the rheology problem near the solution (represented by the dot) produced by GPS

Four out of the seven solutions produced by MADS with 1000 function evaluations (125 to select the initial point and 875 used within the MADS algorithm) are comparable or better to the one produced by GS with  $8 \times 10^9$  function calls.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 8.1.** Select a mesh size parameter  $\delta^k > 0$  and frame size parameter  $\Delta^k \geq \delta^k$ . Let  $M^k$  be the mesh generated by  $\mathbb{D}$  centred at  $x^k$  of coarseness  $\delta^k$ , and  $F^k$  be the frame generated by  $\mathbb{D}$  centred at  $x^k$  of extent  $\Delta^k$ . Let  $P_{\text{GPS}}^k$  be the poll set used by GPS with  $\mathbb{D}_{\text{GPS}}^k = \mathbb{D}$ .

- Prove that  $P_{\text{GPS}}^k \subseteq F^k$ .
- Prove that  $F^k = \{x^k + \delta^k d : d = Dy, y \in \mathbb{N}^p, \delta^k \|d\|_\infty \leq \Delta^k b\}$  where  $D$  is the matrix formed by the columns of  $\mathbb{D}$  and  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$ .

**EXERCISE 8.2.** Suppose the frame size parameter equals  $\Delta^k = 2^{-p}$  and the mesh size parameter equals  $\delta^k = 4^{-p}$  for some positive integer  $p$ .

- a) The matrix

$$D = \begin{bmatrix} 1 & 0 & 1 & 1 & -1 & 0 & -1 & -1 \\ 0 & 1 & 1 & -1 & 0 & -1 & -1 & 1 \end{bmatrix},$$

is used to generate a mesh and a frame centred at  $x^k$ .

Prove that the frame contains  $(2^{p+1} + 1)^2$  points, and therefore  $(2^{p+1} + 1)^2 - 1$  potential polling directions.

- b) Suppose that the columns of  $D$  are the nonzero vectors of the set  $\{-1, 0, 1\}^n$ . How many potential polling directions are there?

**EXERCISE 8.3.** Suppose  $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$  where  $\Delta^k \geq 0$ . Prove  $\Delta^k = \max\{\delta^k, \sqrt{\delta^k}\}$ .

**EXERCISE 8.4.** Let  $\{x^k\}$  be the sequence of iterates produced by applying the MADS algorithm to a function  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with bounded level sets. Prove the following.

- a) There exists at least one limit point of the iteration sequence  $\{x^k\}$ .
- b) If  $f \in \mathcal{C}^0$ , then every limit point has the same function value.

[This proves that Proposition 7.5 still holds when GPS is replaced by MADS.]

**EXERCISE 8.5.** Give an example of a function  $f \in \mathcal{C}^1$ , a closed convex set  $\Omega$ , and point  $\hat{x}$ , such that

$$\hat{x} \in \operatorname{argmin}_x \{f(x) : x \in \Omega\}, \text{ but } \nabla f(\hat{x}) \neq 0.$$

Show that if  $\hat{x}$  is a refined point of the MADS algorithm applied to your example, then it is impossible that the refining directions form a positive spanning set.

**EXERCISE 8.6.** + Show that if  $\hat{x}$  is a refined point (with sets  $L \subseteq K$ ) and if  $d \in T_\Omega^H(\hat{x})$  is a refined direction then  $x^k + \Delta^k d^k \in \Omega$  for all sufficiently large  $k \in L$ .

**EXERCISE 8.7.** Let  $\Omega \subseteq \mathbb{R}^n$  be a nonempty open and bounded set.

- a) Prove that the set  $\{x \in \Omega : x_i \in \mathbb{Z}, i = 1, 2, \dots, n\}$  is finite.
- b) Prove that, given any  $m \in \mathbb{N}$ , the set  $\{x \in \Omega : 10^m x_i \in \mathbb{Z}, i = 1, 2, \dots, n\}$  is finite.
- c) Explain how question b) can be used to create a dense set in  $\Omega$ .
- d) What goes wrong with this approach for the set  $\Omega = \{\sqrt{2}\} \in \mathbb{R}$ ?

**EXERCISE 8.8.** + Explain how to create a set that is an asymptotically dense set of directions in  $\mathbb{R}^n$ .

[Hint, consider the interior of the unit sphere  $\{x \in \mathbb{R}^n : \|x\| < 1\}$  and use techniques similar to Exercise 8.7.]

**EXERCISE 8.9.** For each of the following vectors find the associated Householder matrix, and prove that the columns form an orthonormal basis of  $\mathbb{R}^n$ .

- a)  $v = \frac{1}{5}[-4, -3]^\top$ .
- b)  $v = \frac{1}{13}[-3, -4, 12]^\top$ .
- c)  $v = \frac{1}{\sqrt{26}}[-2, -2, 3, 3]^\top$ .

**EXERCISE 8.10.**  $\square$  Write a program that takes a nonzero vector as an input and outputs the associated Householder matrix. Test your program on the vectors in Exercise 8.9.

**EXERCISE 8.11.** + Prove that given any nonzero normalised vector  $v$  in  $\mathbb{R}^n$ , the columns of the associated Householder matrix will form an orthonormal basis of  $\mathbb{R}^n$ .

**EXERCISE 8.12.**  $\square$  Implement algorithm 8.2. Test it using the following inputs.

- a)  $\delta^k = \frac{1}{64}, \Delta^k = \frac{1}{8}, v^k = \frac{1}{\sqrt{70}}[-5, 3, 6, 0]^\top$
- b)  $\delta^k = \frac{1}{4}, \Delta^k = \frac{1}{2}, v^k = [-4, 3]^\top$
- c)  $\delta^k = \frac{1}{4}, \Delta^k = \frac{1}{2}, v^k = [-3, -4, 12]^\top$
- d)  $\delta^k = \frac{1}{16}, \Delta^k = \frac{1}{4}, v^k = \frac{1}{\sqrt{70}}[-5, 3, 6, 0]^\top$

**EXERCISE 8.13.** + Consider a MADS algorithm using

$$D = [I \ -\mathbf{1}] = \begin{bmatrix} 1 & 0 & \dots & 0 & -1 \\ 0 & 1 & \dots & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

and initial mesh and frame size parameters  $\delta^0 = 1$  and  $\Delta^0 = 1$ . In this question we explore some alternatives to the rule  $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ .

a) Set

$$\delta^{k+1} = \begin{cases} \frac{1}{4}\delta^k & \text{on unsuccessful iterations,} \\ 4\delta^k & \text{on successful iterations with } \delta^k < 1, \quad \text{and } \Delta^{k+1} = 2\delta^{k+1}. \\ 1 & \text{on successful iterations with } \delta^k = 1, \end{cases}$$

Write  $\delta^k$  as a function of  $\Delta^k$ , and describe what happens to the number of possible poll directions after a successful iteration.

b) Set

$$\delta^{k+1} = \begin{cases} \frac{1}{4}\delta^k & \text{on unsuccessful iterations,} \\ \delta^k & \text{on successful iterations,} \end{cases}$$

and

$$\Delta^{k+1} = \begin{cases} \frac{1}{2}\Delta^k & \text{on unsuccessful iterations} \\ \Delta^k & \text{on successful iterations,} \end{cases}$$

Write  $\delta^k$  as a function of  $\Delta^k$ , and describe what happens to the number of possible poll directions after a successful iteration :  $\Delta^{k+1} = 2\Delta^k$ .

c) Explain the advantages and disadvantages of each of the two rules above.

**EXERCISE 8.14.** + This exercise proposes an instance of MADS on an unconstrained convex problem in  $\mathbb{R}^2$  that produces exactly two refined points,  $\mathbf{1} = [1, 1]^\top$  and  $-\mathbf{1} = [-1, -1]^\top$ , and the set of refined directions is not dense for neither of them. Furthermore, at  $-\mathbf{1}$  the descent directions are located in the quadrant  $\mathbb{R}_+^2$ , and when the incumbent solution is near  $-\mathbf{1}$ , then there are no poll directions in the interior of  $\mathbb{R}_+^2$ . A similar behaviour holds for  $\mathbf{1}$ .

Consider the MADS algorithm with parameters  $\Delta^0 = \frac{1}{2}, D = [I - I], \tau = \frac{1}{2}$  applied to the convex function of two variables  $f(x) = \|x\|_\infty$  from Example 3.3 using the starting point  $x^0 = [2, 2]^\top$  and with the following poll direction sets at iteration  $k$ :

$$\mathbb{D}_\Delta^k = \left\{ \pm \frac{\Delta^k}{\delta^k} e_i : i \in \{1, 2\} \right\} \cup \{d^k\},$$

where  $d^k$  is the direction obtained by rounding  $\Delta^k[\cos(k), \sin(k)]^\top$  on the mesh  $M^k$  (the trigonometric functions are in radians). The search set is empty when the iteration number  $k$  is not a multiple of three, and otherwise:

$$S^k = \begin{cases} \{-(1 + \Delta^k)\mathbf{1}\} & \text{if } \{d^{k+1}, d^{k+2}\} \cap \text{int}(\mathbb{R}_+^2) = \emptyset, \\ \{(1 + \Delta^k)\mathbf{1}\} & \text{otherwise.} \end{cases}$$

- a) Show that if  $x^k = \pm(1 + 2^{-\ell})\mathbf{1}$  and  $\delta^k = (\Delta^k)^2 = 4^{-\ell-1}$ , then the search points  $-(1 + \Delta^k)\mathbf{1}$  and  $(1 + \Delta^k)\mathbf{1}$  belong to the mesh  $M^k$ .
- b) Show that if  $\{d^{k+1}, d^{k+2}\} \cap \text{int}(\mathbb{R}_+^2) \neq \emptyset$ , then  $\{d^{k+1}, d^{k+2}\} \cap \text{int}(-\mathbb{R}_+^2) = \emptyset$ . Conclude that if the iteration number  $k$  is not a multiple of three, then the direction  $d^k$  is not a descent direction for  $f$ .
- c) Prove that the sequence of iterations satisfies the following:

| Iteration<br>$k$ | incumbent<br>$x^k$               | parameters<br>$\delta^k$ $\Delta^k$ | search step   | poll step |
|------------------|----------------------------------|-------------------------------------|---|-----------|
| $3\ell$          | $\pm(1 + 2^{-\ell})\mathbf{1}$   | $4^{-\ell-1}$ $2^{-\ell-1}$         | success at<br>$x^{k+1} = \pm(1 + \Delta^k)\mathbf{1}$ | none      |
| $3\ell + 1$      | $\pm(1 + 2^{-\ell-1})\mathbf{1}$ | $4^{-\ell}$ $2^{-\ell}$             | none  | fail      |
| $3\ell + 2$      | $\pm(1 + 2^{-\ell-1})\mathbf{1}$ | $4^{-\ell-1}$ $2^{-\ell-1}$         | none  | fail      |

- d) Prove that the union of all normalised poll directions is dense in the unit sphere, but the union of all normalised refining directions at a refined point is not dense in the unit sphere.
- e) Conclude that there are exactly two refined points:  $\mathbf{1}$  and  $-\mathbf{1}$  and that both of them possess descent directions.



**Chapter 8 Project: Apply MADS to the Nonsmooth Rheology Problem.**

The objective of this project is to implement the most rudimentary MADS algorithm. The search step is void,  $S^k = \emptyset$ ,  $\Delta^0 = 1$ ,  $D = [I - I]$  and  $\tau = \frac{1}{2}$ .

- a) Code a function called `RANDV()` that returns a random unit vector in  $\mathbb{R}^n$  using the 2-norm.
- b) Code algorithm 8.2 (creating the set of poll directions) that takes two inputs: the frame and mesh size parameters. The vector in  $\mathbb{R}^n$  is generated by calling your function `RANDV()`. The algorithm returns the set of  $2n$  polling directions.
- c) Code the MADS algorithm without step 2.
- d) Run your code on the unconstrained nonsmooth Rheology problem from Example 3.1 with different seeds. This will affect the vector returned by `RANDV()`. Discuss your results, and compare with those reported in Example 8.5.

# *Further Remarks on Direct Search Methods*

---

Research in direct search methods has arguably been proceeding since Fermi and Metropolis's 1952 report "Numerical Solution of a Minimization Problem" [68]. It has certainly been proceeding since Hooke and Jeeves's 1961 paper [99]. Since then, numerous researchers have made theoretical advances and practical implementations within the field. It is impossible to list all of the key contributions here. In 2003, Kolda, Lewis, and Torczon [109] published a 100-page survey, which contains an enormous quantity of information on direct search methods. A more recent survey is found in [10]. High-level and historical papers on direct search methods appear in [16, 117, 118, 160, 161].

Positive bases were introduced in 1954 by Davis [57] and some of their properties were recently exposed in [94, 148]. Lewis and Torczon [116] proposed their use within GPS methods. These notions might be more accurately called nonnegative span and nonnegative linearly independent, but historical precedence has created these terms. The upper bound on their cardinality (Theorem 6.2) can be derived using arguments from linear programming as follows.

**PROOF.** (adapted from [9, Thm 2.1]) Let  $\mathbb{D} = \{d_1, d_2, \dots, d_p\}$  be a positive spanning set for  $\mathbb{R}^n$ . Then there exists a basis  $\mathbb{D}' = \{d_j : j \in J\} \subset \mathbb{D}$  of  $\mathbb{R}^n$  (with  $J \subset \{1, 2, \dots, p\}$  and  $|J| = n$ ), otherwise all the columns of  $\mathbb{D}$  would be in the same subspace. Theorem 6.4 shows that the set  $\mathbb{D}' \cup b$  is a positive basis, when  $b = -\sum_{j \in J} d_j$ . Now, since  $\mathbb{D}$  is a positive spanning set, it follows that the set  $\{x \in \mathbb{R}^p : \sum_{i=1}^p d_i x_i = b, x \geq 0\}$  is nonempty. Linear programming theory ensures the existence of an  $\bar{x} \geq 0$  in  $\mathbb{R}^n$  such that  $\sum_{i=1}^p d_i \bar{x}_i = b$  and  $\bar{x}_i = 0$  for at least  $p - n$  indices. It follows that the cardinality of the set  $I = \{i \in \{1, 2, \dots, p\} : \bar{x}_i \neq 0\}$  is at most  $n$ .

The set  $\mathbb{B} = \{d_j : j \in I \cup J\} \subseteq \mathbb{D}$  positively spans  $\mathbb{D}'$  as well as  $b$  (since  $\sum_{i \in I} d_i \bar{x}_i = b$  and  $\bar{x} \geq 0$ ). It follows that the subset  $\mathbb{B}$  positively spans the positive basis  $\mathbb{D}' \cup \{b\}$ . Finally, since  $|\mathbb{B}| \leq |I| + |J| \leq 2n$ , a positive basis has at most  $2n$  elements.  $\square$

Positive bases are also used by the so-called *implicit filtering* sampling method which improve on CS by constructing a local model of the objective function using quasi-Newton methods. In the book [105], Kelley presents the algorithm, its supporting convergence theory, a MATLAB implementation [84] named IFFCO, and many case studies. Implicit filtering is applied to many engineering problems in [40, 77, 85, 170].

Torczon [159] introduced the GPS class of algorithms that encompass CS [68], *evolutionary operation* using factorial design [36], the original *pattern search algorithm* [99], and the *multidirectional search algorithm* [58].

In order to simplify the book's presentation, the mesh size update rules were to either multiply or divide by the rational number  $\tau$ . The original GPS and MADS algorithms allow to multiply by a bounded positive power of  $\tau$  and divide by a nonnegative power of  $\tau$ . Analysis of GPS with positive bases and  $\mathcal{C}^2$  functions was done in [49, 115, 173]. A connection between the mesh size parameter and the norm of the gradient is exposed in [64].

The theoretical analysis of GPS is improved using the Clarke nonsmooth calculus [41] in [13]. Limits of the convergence analysis are exposed in [8]. These convergence analyses rely on the notion of the mesh. Observe that due to the multiplication by the real matrix  $G$ , the requirement that  $Z$  is integer is equivalent to assuming that  $Z$  is rational. The mesh is substituted by a minimal decrease of the objective function requirement in [80, 122] as well as in *frame-based* methods [48, 142] and in *generating set search* methods [109, 110].

Part iii. of Theorem 7.7 on the convergence of GPS shows that if  $f \in \mathcal{C}^1$ , then the gradient is null. This result can be strengthen to “if  $f$  is differentiable and regular at  $x$ , then the gradient is null” [13]. Functions in the intersection of differentiability and regularity, as seen in Figure 6.2, are said to be strictly differentiable [114]. Strict differentiability implies that the generalised gradient from Definition 6.8 is the singleton:  $\partial f(x) = \{\nabla f(x)\}$ .

The MADS algorithm was introduced in [15], as a direct search method for constrained problems. The hierarchical convergence was derived using the nonsmooth calculus elements from the books of Clarke [41], Hiriart-Urruty and Lemaréchal [97], and Jahn [102]. The analysis involves additional types of tangent cones. A similar convergence analysis is developed for the DIRECT algorithm [69] and to sampling methods [71]. A result for non-Lipschitz functions is presented in [165] using the Rockafellar upper subderivative [151]. Second order analyses are developed in [2, 3]. Alternative techniques to generate the MADS polling directions are proposed in [6, 15, 153, 163].

The most complete implementation of the MADS algorithm is the Nomad software package [5, 20, 112] written in C++. In addition to the MADS algorithm, it includes the LHS, CS, GPS direct search algorithms, and all functionalities described in Part 5 of the present book.

PART

# 4

## Model-Based Methods

Part 4 presents two types of model-based methods for derivative-free unconstrained optimization problems.

- Chapter 9 shows how to construct linear and quadratic models, even when there are not enough or too many sample points.
- Chapter 10 describes model-based descent methods, which generalise a classic first-order method (gradient based descent) to use approximate gradients. In this method, convergence is proven without the need to drop to subsequences, accumulation points, or refined points.
- Chapter 11 describes model-based trust-region methods, which generalise traditional trust region methods from smooth optimization to the derivative-free optimization setting.

Conclusion: if models are of sufficient quality, then it is possible to adapt algorithms from smooth optimization to derivative-free optimization.

# *Building Linear and Quadratic Models*

Through this book, and in all areas of optimization, it should always be remembered that whenever derivatives are available, they should be used. Derivatives, or more generally gradients, provide powerful information about a function. In unconstrained optimization, descent directions and first order optimality hinge on these basic objects of multivariate calculus. Hessian matrices provide extremely powerful information regarding the local curvature of a function, and can be employed to build quadratically convergent algorithms.

Of course, in the field of DFO we are working under the assumption that derivatives are not readily available. However, in many cases it is reasonable to assume that the objective function is smooth, or locally smooth. For example, if the objective function is known to be a numerical approximation of an integral, then it is reasonable to assume differentiability. In situations like this, it may be valuable to approximate the objective function with a model function, and use the gradients or even second derivatives of the model function to help guide optimization. This is referred to as a model-based method.

Model-based methods are popular in DFO, as they provide interesting mathematical theory, useful stopping information, and have been found effective in many real-world applications.

Chapters 9, 10, and 11 study model-based DFO methods for unconstrained minimisation of a smooth function. In this chapter we focus on the question of how to build “good” models. These constructions will be applied in Chapters 10 and 11, to create model-based DFO algorithms. To stay true to the DFO setting, each model is constructed using only function values computed at carefully selected sample points. The first class of models consists of building a linear function in  $\mathbb{R}^n$  that interpolates through exactly  $n+1$  points defining a simplex. The second class of models generalises this by considering more than  $n+1$  points and using linear regression. Finally, we examine methods that build quadratic models using once again interpolation techniques.

Before we examine the model construction techniques, we must define when models are sufficiently “good” to be of use.

### 9.1. Fully Linear Models

Ideally, good models should be easy to build, at least in an algorithmic sense, and easy to work with, i.e., easy to minimise or easy to extract gradient and Hessian information from. More importantly, good models should be similar enough to the true function to provide useful information. To formalise this notion, we introduce the idea of a *class of fully linear models*.

**DEFINITION 9.1 (Class of Fully Linear Models).**

Given  $f \in \mathcal{C}^1$ ,  $x \in \mathbb{R}^n$ , and  $\bar{\Delta} > 0$  we say that  $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$  is a class of fully linear models of  $f$  at  $x$  parameterised by  $\Delta$  if there exists a pair of scalars  $\kappa_f(x) > 0$  and  $\kappa_g(x) > 0$  such that, given any  $\Delta \in (0, \bar{\Delta}]$  the model  $\tilde{f}_\Delta$  satisfies

$$\begin{aligned} |f(y) - \tilde{f}_\Delta(y)| &\leq \kappa_f(x)\Delta^2 && \text{for all } y \in B_\Delta(x) \\ \text{and } \|\nabla f(y) - \tilde{g}_\Delta(y)\| &\leq \kappa_g(x)\Delta && \text{for all } y \in B_\Delta(x) \end{aligned}$$

where  $\tilde{g}_\Delta = \nabla \tilde{f}_\Delta$ .

In a class of fully linear models at some given  $x \in \mathbb{R}^n$ , the parameters  $\kappa_f(x)$  and  $\kappa_g(x)$  should be taken as unknown but constant, while the parameter  $\Delta$  should be taken as under the control of the optimizer. Thus, the idea behind having a class of fully linear models at  $x$  is that, while the optimizer may not know the exact level of error the model contains, the optimizer has access to a parameter ( $\Delta$ ) which can be used to drive the error to 0 in a predictable way.

The bounds  $(0, \bar{\Delta}]$  play an important role in proving that classes of fully linear models exist. The lower bound  $\Delta > 0$  is strict, as selecting  $\Delta = 0$  has no practical interpretation. The upper bound  $\Delta \leq \bar{\Delta}$  is required to ensure, under reasonable situations, that the pair of scalars  $\kappa_f(x) > 0$  and  $\kappa_g(x) > 0$  are independent of  $\Delta$ . This will be discussed further in Sections 9.2

through 9.5, where methods for constructing classes of fully linear models are developed.

We use the following shorthand to represent situations where the optimizer can create a class of fully linear models of  $f$  at any  $x \in \mathbb{R}^n$ . The last part of the definition introduces the case where the pair of scalars  $\kappa_f(\cdot)$  and  $\kappa_g(\cdot)$  do not depend on the value of  $x$ .

**DEFINITION 9.2** (Fully Linear Models).

Let  $f \in \mathcal{C}^1$ .

We say that  $\tilde{f}_\Delta$  are fully linear models of  $f$  to mean that, given any  $x \in \mathbb{R}^n$  and  $\bar{\Delta} > 0$ , there exists a class  $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$  of fully linear models of  $f$  at  $x$  parameterised by  $\Delta$ .

We say that  $\tilde{f}_\Delta$  are fully linear models of  $f$  with constants  $\kappa_f$  and  $\kappa_g$  to mean that, given any  $x \in \mathbb{R}^n$  and  $\bar{\Delta} > 0$ , there exists a class  $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$  of fully linear models of  $f$  at  $x$  parameterised by  $\Delta$  and the scalars  $\kappa_f(x)$  and  $\kappa_g(x)$  can be taken as the constants:  $\kappa_f(x) = \kappa_f$  and  $\kappa_g(x) = \kappa_g$  (independent of  $x$ ).

The next example illustrates the notion of fully linear models on a function of a single variable.

**EXAMPLE 9.1.** Consider the single-variable function  $f(x) = \sin(\pi x)$  at the point  $\bar{x} = 1$ .

The first-order Taylor approximation of  $f$  at  $\bar{x}$  is

$$T(y) = f(\bar{x}) + \frac{d}{dx}f(\bar{x})(x - \bar{x}) = -\pi(x - 1).$$

Fix  $\Delta > 0$ . Given any  $y \in [1 - \Delta, 1 + \Delta]$ , Taylor's Theorem ensures that the error in this model satisfies

$$\begin{aligned} |f(y) - T(y)| &\leq \frac{1}{2} \max_{\xi \in [1 - \Delta, 1 + \Delta]} \left| \frac{d^2}{dx^2} f(\xi) \right| |y - 1|^2 \\ &\leq \frac{1}{2} \max_{\xi \in [1 - \Delta, 1 + \Delta]} |\pi^2 \sin(\pi\xi)| |y - 1|^2 \leq \frac{1}{2} \pi^2 \Delta^2, \end{aligned}$$

as  $|y - 1| \leq \Delta$ .

Now examine the first-order error in this model,

$$\left| \frac{d}{dx} f(y) - \frac{d}{dx} T(y) \right| = |\pi \cos(\pi y) - (-\pi)| = |\pi \cos(\pi y) - \pi \cos(\pi)|.$$

Considering the function  $g(x) = \pi \cos(\pi x)$ , we see that  $\pi \cos(\pi)$  is the zeroth-order Taylor expansion of  $g$  at  $\bar{x}$ . Again, applying Taylor's Theorem, given any  $y \in [1 - \Delta, 1 + \Delta]$ , the error in this model satisfies

$$\begin{aligned} (9.1) \quad |\pi \cos(\pi y) - \pi \cos(\pi)| &\leq \max_{\xi \in [1 - \Delta, 1 + \Delta]} \left| \frac{d}{dx} g(\xi) \right| |y - 1| \\ &\leq \max_{\xi \in [1 - \Delta, 1 + \Delta]} |\pi^2 \sin(\pi\xi)| |y - 1| \leq \pi^2 \Delta. \end{aligned}$$

Thus, the model  $T$  is fully linear with constants  $\kappa_f = \frac{1}{2}\pi^2$  and  $\kappa_g = \pi^2$ .

In reviewing Example 9.1, it should be clear that the only special property of  $f(x) = \sin(\pi x)$  is that  $f \in \mathcal{C}^2$  (which allowed for the application of Taylor's Theorem). In fact, if any  $f \in \mathcal{C}^{1+}$  on  $B_{\Delta}(x)$ , then the first-order Taylor approximation will create a fully linear model of  $f$  at  $x$  (see Exercise 9.1).

Another example of constructing fully linear models of a single-variable function comes from the secant line.

**EXAMPLE 9.2.** Consider the single-variable function  $f(x) = \sin(\pi x)$  at the point  $\bar{x} = 1$ . Fix  $\Delta > 0$  and define the secant model

$$S(y) = f(\bar{x}) + \left( \frac{f(\bar{x} + \Delta) - f(\bar{x})}{\Delta} \right) (y - \bar{x}).$$

By the Mean Value Theorem, there exists  $\varsigma \in (\bar{x}, \bar{x} + \Delta)$  such that

$$\frac{d}{dx} f(\varsigma) = \frac{f(\bar{x} + \Delta) - f(\bar{x})}{\Delta}.$$

Thus,

$$\begin{aligned} S(y) &= f(\bar{x}) + \frac{d}{dx} f(\varsigma)(y - \bar{x}) \\ &= f(\bar{x}) + \frac{d}{dx} f(\bar{x})(y - \bar{x}) - \frac{d}{dx} f(\bar{x})(y - \bar{x}) + \frac{d}{dx} f(\varsigma)(y - \bar{x}) \\ &= T(y) + \left( \frac{d}{dx} f(\varsigma) - \frac{d}{dx} f(\bar{x}) \right) (y - \bar{x}), \end{aligned}$$

where  $T$  is the first-order Taylor approximation, examined in Example 9.1.

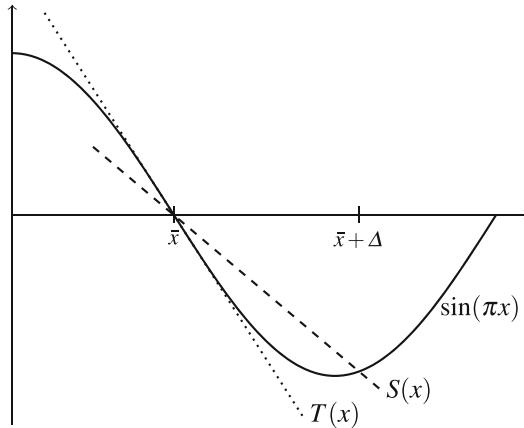


FIGURE 9.1. The first-order Taylor approximation  $T$  and the secant approximation  $S$  both form fully linear models of  $f(x) = \sin(\pi x)$

Figure 9.1 visualises  $S$  beside the first-order Taylor approximation  $T$ .

Notice that

$$\left| \left( \frac{d}{dx} f(\varsigma) - \frac{d}{dx} f(\bar{x}) \right) \right| = \left| \pi \cos(\pi \varsigma) - \pi \cos(\pi) \right| \leq \pi^2 \Delta.$$

by Equation (9.1). From here it is easy to confirm

$$\begin{aligned} |S(y) - f(y)| &\leq |S(y) - T(y)| + |T(y) - f(y)| \leq \pi^2 \Delta^2 + \frac{1}{2}\pi^2 \Delta^2, \text{ and} \\ \left| \frac{d}{dx} S(y) - \frac{d}{dx} f(y) \right| &\leq \left| \frac{d}{dx} S(y) - \frac{d}{dx} T(y) \right| + \left| \frac{d}{dx} T(y) - \frac{d}{dx} f(y) \right| \pi^2 \Delta + \pi^2 \Delta. \end{aligned}$$

Thus the model  $S$  is fully linear with constants  $\kappa_f = \frac{3}{2}\pi^2$  and  $\kappa_g = 2\pi^2$ .

---

Examining Example 9.2, it is again clear that the only special property of  $f$  is that  $f \in \mathcal{C}^2$ . In fact, in Section 9.2, we will generalise the ideas in Example 9.2 to work for  $f : \mathbb{R}^n \mapsto \mathbb{R}$  with  $f \in \mathcal{C}^{1+}$  (although the error bounds will be generalised using Lemma 9.3 rather than the Mean Value Theorem).

In most model-based DFO algorithms it is not necessary to have fully linear models with constants  $\kappa_f$  and  $\kappa_g$ , only that the parameter  $\kappa_f(x)$  and  $\kappa_g(x)$  are locally bounded near any accumulation points of the algorithm. Conceptually, the proofs do not become more difficult when  $\kappa_f(x)$  and  $\kappa_g(x)$  are only locally bounded near any accumulation point, but the proofs do become more “technical” in that situation.

In Chapter 10, we shall see that fully linear models are actually stronger than what we need to ensure convergence of the model-based descent algorithm. While in Chapter 11, we shall use the full power of fully linear models.

The remainder of this chapter introduces four ways to construct classes of fully linear models. First, an interpolation technique to construct a linear function in  $\mathbb{R}^n$  that passes through exactly  $n+1$  points is discussed. Second, when more than  $n+1$  points are available we examine least square regression to find the best linear function that approximates the function. Third and fourth, again in the situation where there are more than  $n+1$  points, we examine two interpolation methods to construct quadratic models.

## 9.2. Linear Interpolation

Fully linear models require a model  $\tilde{f}_\Delta$  of the objective function  $f$ , whose gradient is an approximation of the true gradient  $\nabla f$ , when it exists. Thus, it should be immediately clear that the model should be at least sophisticated enough to have a nontrivial gradient. As we desire the model to be easily differentiable, an obvious first choice for such a model would be a linear function

$$\tilde{f}_\Delta(x) = L(x) = \alpha_0 + \alpha^\top x,$$

where  $\alpha_0 \in \mathbb{R}$ ,  $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]^\top \in \mathbb{R}^n$ . Constructing  $L(x)$  via the Taylor polynomial of degree 1 is out of the question, as it requires knowledge of the unknown gradient  $\nabla f$ .

Consider instead a set  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  of  $m+1$  points. If we treat  $\mathbb{Y}$  as *interpolation points*, then we can build an *interpolation model* by seeking values for  $\alpha^0$  and  $\alpha$  that satisfy

$$L(y^i) = f(y^i) \quad \text{for each } y^i \in \mathbb{Y}.$$

Writing this in terms of  $\alpha_0$  and  $\alpha$  we require

$$(9.2) \quad L(y^i) = \alpha_0 + \alpha_1 y_1^i + \alpha_2 y_2^i + \dots + \alpha_n y_n^i = f(y^i) \quad \text{for each } y^i \in \mathbb{Y}.$$

This creates a system of linear equations with  $n + 1$  unknowns ( $\alpha_i$ ,  $i = 0, 1, \dots, n$ ) and  $m + 1$  equations (one for each element of the set  $\mathbb{Y}$ ). As such, we will require (at least)  $n + 1$  interpolation points. For now, let us focus on  $m = n$ , and leave the case where  $m > n$  for a later section. Rewriting Equation (9.2) in matrix form we get

$$\begin{bmatrix} 1 & (y^0)^\top \\ 1 & (y^1)^\top \\ \vdots & \vdots \\ 1 & (y^n)^\top \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^n) \end{bmatrix}.$$

To tighten notation, we define

$$f(\mathbb{Y}) := \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^n) \end{bmatrix} \in \mathbb{R}^{n+1} \quad \text{and} \quad Y := [y^0 \ y^1 \ \dots \ y^n] \in \mathbb{R}^{n \times (n+1)}.$$

So Equation (9.2) may be compactly written as

$$(9.3) \quad [\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y}),$$

where  $\mathbf{1} \in \mathbb{R}^{n+1}$  is the vector of all ones. As we want Equation (9.3) to have a unique solution, we require  $[\mathbf{1} \ Y^\top]$  to be invertible. We use the term *poised* to refer to this situation.

**DEFINITION 9.3 (Poised for Linear Interpolation).**

The set  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$ , is poised for linear interpolation if the  $(n + 1) \times (n + 1)$  matrix  $[\mathbf{1} \ Y^\top]$  is invertible, where  $Y = [y^0 \ y^1 \ \dots \ y^n]$  and  $\mathbf{1} \in \mathbb{R}^{n+1}$  is the vector of all ones.

**DEFINITION 9.4 (Linear Interpolation Function).**

Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  be poised for linear interpolation and  $f : \mathbb{R}^n \mapsto \mathbb{R}$ . Then the linear interpolation function of  $f$  over  $\mathbb{Y}$  is

$$L_Y(x) := \alpha_0 + \alpha^\top x$$

where  $(\alpha_0, \alpha)$  is the unique solution to  $[\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y})$ .

Poised for linear interpolation implies that the system of linear equations  $L_Y(y^i) = \alpha_0 + \alpha^\top y^i$  has a unique solution. In  $\mathbb{R}^2$ , this means that  $\mathbb{Y} = \{y^0, y^1, y^2\} \subset \mathbb{R}^2$  are not collinear. With this in mind, it should be no surprise that poised for linear interpolation is intimately linked with simplices.

**PROPOSITION 9.1.**

Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$ . The following are equivalent.

- i.  $\mathbb{Y}$  is poised for linear interpolation,
- ii.  $\mathbb{Y}$  forms a simplex,
- iii. the matrix  $L = [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0] \in \mathbb{R}^{n \times n}$  is invertible.

**PROOF.** Using  $Y = [y^0 \quad y^1 \quad \dots \quad y^n]$ , we note that

$$\begin{aligned} \mathbb{Y} \text{ is poised for linear interpolation} &\Leftrightarrow \begin{bmatrix} \mathbf{1} & Y^\top \end{bmatrix} \text{ is invertible} \\ \Leftrightarrow \det \begin{bmatrix} 1 & (y^0)^\top \\ 1 & (y^1)^\top \\ \vdots & \vdots \\ 1 & (y^n)^\top \end{bmatrix} \neq 0 &\Leftrightarrow \det \begin{bmatrix} 1 & (y^0)^\top \\ 1-1 & (y^1-y^0)^\top \\ \vdots & \vdots \\ 1-1 & (y^n-y^0)^\top \end{bmatrix} \neq 0 \\ \Leftrightarrow \det \begin{bmatrix} 1 & (y^0)^\top \\ \mathbf{0} & L^\top \end{bmatrix} \neq 0 &\Leftrightarrow L^\top \text{ is invertible} \\ \Leftrightarrow \mathbb{Y} \text{ forms a simplex (by Theorem 2.5).} & \end{aligned}$$

□

Due to the link between being poised for linear interpolation and simplices, the gradient of the linear interpolation function is usually referred to as the *simplex gradient*.

**DEFINITION 9.5 (Simplex Gradient).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  be poised for linear interpolation. The simplex gradient of  $f$  over  $\mathbb{Y}$ , denoted  $\nabla_S f(\mathbb{Y})$ , is the gradient of the linear interpolation function  $L_Y$  of  $f$  over  $\mathbb{Y}$ :

$$\nabla_S f(\mathbb{Y}) := \nabla L_Y(x) = \alpha.$$

The notation for the simplex gradient uses the symbol  $\mathbb{Y}$ , as the simplex gradient is defined through the elements of  $\mathbb{Y}$ . Also, as  $L_Y$  is a linear function, it does not matter where we evaluate the gradient of  $L_Y$ .

Before we show that the simplex gradient creates a class of fully linear models, we provide a simplified formula for computing the simplex gradient from the set  $\mathbb{Y}$  and vector  $f(\mathbb{Y})$ . The simplification results in solving a  $n \times n$  linear system of equations rather than the  $(n+1) \times (n+1)$  system of Definition 9.4.

**PROPOSITION 9.2.**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  be poised for linear interpolation. Define

$$L := [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0] \in \mathbb{R}^{n \times n} \text{ and}$$

$$\delta^{f(\mathbb{Y})} := \begin{bmatrix} f(y^1) - f(y^0) \\ f(y^2) - f(y^0) \\ \vdots \\ f(y^n) - f(y^0) \end{bmatrix} \in \mathbb{R}^n.$$

Then the simplex gradient  $\alpha = \nabla_S f(\mathbb{Y})$  is the unique solution of the linear system

$$L^\top \alpha = \delta^{f(\mathbb{Y})}.$$

**PROOF.** See Exercise 9.2. □

### 9.3. Error Bounds for Linear Interpolation

Our next goal is to show the simplex gradient defines a class of fully linear models. We shall use the parameter  $\Delta$  to control accuracy through the *approximate simplex diameter*. Recall, if  $\mathbb{Y}$  forms a simplex, then the approximate diameter of the simplex  $\mathbb{Y}$  is

$$\overline{\text{diam}}(\mathbb{Y}) = \max\{\|y^1 - y^0\|, \|y^2 - y^0\|, \dots, \|y^n - y^0\|\} \quad (\text{see Definition 2.18}).$$

We seek to show that given  $x \in \mathbb{R}^n$  there exists a bound  $\bar{\Delta} > 0$ , and a pair of scalars  $\kappa_f(x)$  and  $\kappa_g(x) > 0$ , such that if  $\Delta = \overline{\text{diam}}(\mathbb{Y}) \in (0, \bar{\Delta}]$ , then the linear interpolation model satisfies the inequalities

$$\begin{aligned} |f(y) - L_Y(y)| &\leq \kappa_f(x)\Delta^2 && \text{for all } y \in B_\Delta(x) \\ \text{and } \|\nabla f(x) - \alpha\| &\leq \kappa_g\Delta && \text{for all } y \in B_\Delta(x). \end{aligned}$$

Of course, for this to work, we will need to pick  $\mathbb{Y}$  carefully. In particular, we will need to select  $y^0 = x$ . The parameters  $\kappa_g(x)$  and  $\kappa_f(x)$  for the fully linear inequalities will then depend on several variables, including the dimension of the space, the Lipschitz constant of  $\nabla f$ , and the norm of the inverse of the matrix

$$(9.4) \quad \hat{L} := \frac{1}{\overline{\text{diam}}(\mathbb{Y})} [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0].$$

The proof of the error bounds for the simplex gradient requires two lemmas.

**LEMMA 9.3 (Fundamental Theorem of Calculus in  $\mathbb{R}^n$ ).**

Let  $f \in \mathcal{C}^1$  and  $x, d \in \mathbb{R}^n$ . Then

$$f(x + d) - f(x) = \int_0^1 d^\top \nabla f(x + \tau d) d\tau.$$

**PROOF.** This can be derived from applying the fundamental theorem of calculus in  $\mathbb{R}$  to the function  $\psi(\tau) = f(x + \tau d)$ .  $\square$

The next lemma requires Lipschitz continuity of the gradient (see the paragraph following Definition 2.2).

**LEMMA 9.4 (First-Order Taylor Approximation Error Bound).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . For any  $d \in B_{\bar{\Delta}}(0)$

$$|f(x + d) - f(x) - d^\top \nabla f(x)| \leq \frac{1}{2} K \|d\|^2.$$

**PROOF.** Using Lemma 9.3 and properties of the integral gives

$$\begin{aligned} |f(x + d) - f(x) - d^\top \nabla f(x)| &= \left| \int_0^1 d^\top \nabla f(x + \tau d) d\tau - d^\top \nabla f(x)^\top \right| \\ &= \left| \int_0^1 d^\top \nabla f(x + \tau d) - d^\top \nabla f(x) d\tau \right| \\ &= \left| \int_0^1 d^\top (\nabla f(x + \tau d) - \nabla f(x)) d\tau \right| \\ &\leq \int_0^1 |d^\top (\nabla f(x + \tau d) - \nabla f(x))| d\tau \\ &\leq \int_0^1 \|d\| \|\nabla f(x + \tau d) - \nabla f(x)\| d\tau \\ &\leq \int_0^1 K \|d\| \|x + \tau d - x\| d\tau \\ &= K \|d\|^2 \int_0^1 \tau d\tau = \frac{1}{2} K \|d\|^2. \end{aligned}$$

$\square$

At last, we prove that the Simplex Gradient creates a class of fully linear models. The proof is broken into two parts (Theorems 9.5 and Corollary 9.6).

**THEOREM 9.5 (Simplex Gradient Error Bound).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y}$  be poised for linear interpolation with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}) \leq \bar{\Delta}$ . Then the simplex gradient  $\alpha = \nabla_S f(\mathbb{Y})$  satisfies

$$(9.5) \quad \|\nabla f(y) - \alpha\| \leq \left( \frac{1}{2} K \sqrt{n} \|\hat{L}^{-1}\| \right) \Delta,$$

where  $\hat{L}$  is the matrix from Equation (9.4). Furthermore,

$$(9.6) \quad \|\nabla f(y) - \alpha\| \leq \left( \frac{1}{2} K (2 + \sqrt{n} \|\hat{L}^{-1}\|) \right) \Delta \quad \text{for all } y \in B_\Delta(x).$$

**PROOF.** Let  $L_Y(x) = \alpha_0 + \alpha^\top x$  be the linear interpolation function of  $f$  over  $\mathbb{Y}$ . Using Lemma 9.4, for any  $i = 1, 2, \dots, n$ , we have

$$\begin{aligned} \frac{1}{2}K\|y^i - y^0\|^2 &\geq |f(y^i) - f(y^0) - (y^i - y^0)^\top \nabla f(y^0)| \\ &= |\alpha^\top (y^i - y^0) - (y^i - y^0)^\top \nabla f(y^0)| \quad \text{since } f(y^j) = L_Y(y^j) \\ &= |(y^i - y^0)^\top (\alpha - \nabla f(y^0))|. \end{aligned}$$

With this established, consider the quantity

$$\widehat{L}^\top(\alpha - \nabla f(y^0)) = \frac{1}{\Delta} \begin{bmatrix} (y^1 - y^0)^\top \\ (y^2 - y^0)^\top \\ \vdots \\ (y^n - y^0)^\top \end{bmatrix} (\alpha - \nabla f(y^0)),$$

and in particular, the square of its norm satisfies

$$\begin{aligned} \|\widehat{L}^\top(\alpha - \nabla f(y^0))\|^2 &= \frac{1}{\Delta^2} \sum_{i=1}^n ((y^i - y^0)^\top (\alpha - \nabla f(y^0)))^2 \\ &\leq \frac{1}{\Delta^2} \sum_{i=1}^n \left( \frac{1}{2}K\|y^i - y^0\|^2 \right)^2 \\ &\leq \frac{n}{4}K^2\Delta^2 \quad \text{since } \|y^i - y^0\| \leq \Delta. \end{aligned}$$

Noting that  $\|A\| = \|A^\top\|$ , we find that

$$\begin{aligned} \|\alpha - \nabla f(y^0)\| &= \|\widehat{L}^{-\top}\widehat{L}^\top(\alpha - \nabla f(y^0))\| \\ &\leq \|\widehat{L}^{-\top}\| \|\widehat{L}^\top(\alpha - \nabla f(y^0))\| \\ &\leq \left( \frac{1}{2}K\sqrt{n}\|\widehat{L}^{-1}\| \right) \Delta, \end{aligned}$$

which is Equation (9.5). To see Equation (9.6), note whenever  $y \in B_\Delta(y^0)$  we have

$$\begin{aligned} \|\nabla f(y) - \alpha\| &\leq \|\alpha - \nabla f(y^0)\| + \|\nabla f(y^0) - \nabla f(y)\| \\ &\leq \left( \frac{1}{2}K\sqrt{n}\|\widehat{L}^{-1}\| \right) \Delta + K\|y^0 - y\| \\ &\leq \left( \frac{1}{2}K\sqrt{n}\|\widehat{L}^{-1}\| \right) \Delta + K\Delta. \end{aligned}$$

□

Notice Theorem 9.5 requires a fixed Lipschitz constant  $K$  for the gradient of the function  $f$  on  $B_{\bar{\Delta}}(x)$ . Without the bounding ball  $B_{\bar{\Delta}}(x)$ , this condition would be unreasonable, as even “simple” functions such as  $f(x) = e^x$  would not satisfy the condition (see Exercise 9.6 for more discussion).

As a corollary, we see that linear interpolation creates a class of fully linear models.

**COROLLARY 9.6 (Linear Interpolation Is Fully Linear).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y}$  be poised for linear interpolation with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}) \leq \bar{\Delta}$ . Then the linear interpolation function  $L_Y$  of  $f$  over  $\mathbb{Y}$  satisfies

(9.7)

$$|f(y) - L_Y(y)| \leq \left( \frac{1}{2} K (1 + \sqrt{n} \|\hat{L}^{-1}\|) \right) \Delta^2 \quad \text{for all } y \in B_\Delta(x),$$

where  $\hat{L}$  is the matrix from Equation (9.4).

**PROOF.** Using  $L_Y(y^0) = f(y^0)$  and  $L_Y(y) - L_Y(y^0) = \alpha^\top (y - y^0)$  we have

$$\begin{aligned} |f(y) - L_Y(y)| &= |f(y) - f(y^0) + L_Y(y^0) - L_Y(y)| \\ &= |f(y) - f(y^0) + \alpha^\top (y^0 - y) \\ &\quad + (y - y^0)^\top \nabla f(y^0) - (y - y^0)^\top \nabla f(y^0)| \\ &\leq |f(y) - f(y^0) - (y - y^0)^\top \nabla f(y^0)| + \|y - y^0\| \|\nabla f(y^0) - \alpha\|. \end{aligned}$$

Applying Lemma 9.4, Theorem 9.5, and  $\|y - y^0\| \leq \Delta$  yields

$$\begin{aligned} |f(y) - L_Y(y)| &\leq \frac{1}{2} K \|y - y^0\|^2 + \|y - y^0\| \left( \frac{1}{2} K \sqrt{n} \|\hat{L}^{-1}\| \right) \Delta \\ &\leq \frac{1}{2} K \Delta^2 + \left( \frac{1}{2} K \sqrt{n} \|\hat{L}^{-1}\| \right) \Delta^2. \end{aligned} \quad \square$$

Using Theorem 9.5 and Corollary 9.6, we see that linear interpolation creates a class of fully linear models with parameters

$$\kappa_f = \left( \frac{K}{2} (1 + \sqrt{n} \|\hat{L}^{-1}\|) \right) \quad \text{and} \quad \kappa_g = \left( \frac{K}{2} (2 + \sqrt{n} \|\hat{L}^{-1}\|) \right).$$

These parameters depend on  $\|\hat{L}^{-1}\|$ , which in turn depends on the sample set  $\mathbb{Y}$ . As such, it may not be obvious that we can drive  $\Delta$  to 0 while keeping  $\|\hat{L}^{-1}\|$  bounded above. We end this section with a lemma showing that  $\|\hat{L}^{-1}\|$  is easily controlled.

**PROPOSITION 9.7.**

Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  be poised for linear interpolation. Given  $\gamma > 0$ , for  $i = 0, 1, \dots, n$  define  $y_\gamma^i := y^0 + \gamma(y^i - y^0)$ . Then

$$\mathbb{Y}_\gamma := \{y_\gamma^0, y_\gamma^1, \dots, y_\gamma^n\}$$

is also poised for linear interpolation. If

$$\hat{L} = \frac{1}{\overline{\text{diam}}(\mathbb{Y})} [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0] \quad \text{and}$$

$$\hat{L}_\gamma = \frac{1}{\overline{\text{diam}}(\mathbb{Y}_\gamma)} [y_\gamma^1 - y_\gamma^0 \quad y_\gamma^2 - y_\gamma^0 \quad \dots \quad y_\gamma^n - y_\gamma^0],$$

then  $\overline{\text{diam}}(\mathbb{Y}_\gamma) = \gamma \overline{\text{diam}}(\mathbb{Y})$  and  $\|\hat{L}^{-1}\| = \|\hat{L}_\gamma^{-1}\|$ .

**PROOF.** See Exercise 9.3. □

## 9.4. Linear Regression

While the simplex gradient provides a quick, easy, and relatively effective method of obtaining approximate gradients, it is by no means the only approach. In practice, the DFO algorithms evaluate the objective function value at many trial points, and at a given iteration, it is quite possible that the region delimited by the radius  $\Delta$  contains more than  $n + 1$  evaluated points, i.e., more points than what is necessary to form a simplex in  $\mathbb{R}^n$ . In this case, it seems a shame to throw information away. One method of dealing with the additional information is to replace the linear interpolation model with a least squares linear regression model. The next two definitions are very similar to Definitions 9.3 and 9.4, but notice the number of points has changed from a strict  $n + 1$  to an arbitrary  $m + 1 \geq n + 1$ .

**DEFINITION 9.6** (Poised for Linear Regression).

The set  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  with  $m \geq n$  is poised for linear regression if the rank of the  $(m+1) \times (n+1)$  matrix  $[\mathbf{1} \ Y^\top]$  is  $n+1$ , where  $Y = [y^0 \ y^1 \ \dots \ y^m] \in \mathbb{R}^{n \times (m+1)}$  and  $\mathbf{1} \in \mathbb{R}^{m+1}$  is the vector of all ones.

**DEFINITION 9.7** (Linear Regression Function).

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  with  $m \geq n$ , be poised for linear regression. Then the linear regression function of  $f$  over  $\mathbb{Y}$  is

$$L_Y(x) := \alpha_0 + \alpha^\top x$$

where  $(\alpha_0, \alpha)$  is the unique solution to the least square problem

$$(9.8) \quad \min_{\alpha_0, \alpha} \left\{ \sum_{i=0}^m (\alpha_0 + \alpha^\top y^i - f(y^i))^2 \right\}.$$

Notice that since the  $y^i$  are fixed, Equation (9.8) minimises the sum of squares of linear functions. Using  $Y$  as in Definition 9.6 and  $f(\mathbb{Y}) := [f(y^0) \ f(y^1) \ \dots \ f(y^m)]^\top$ , the objective function of this minimisation problem may be rewritten in matrix form as

$$\phi(\alpha_0, \alpha) := \left\| [\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} - f(\mathbb{Y}) \right\|^2.$$

To simplify the analysis, let us introduce the notation:

$$a = \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} \in \mathbb{R}^{n+1}, \quad M = [\mathbf{1} \ Y^\top] \in \mathbb{R}^{(m+1) \times (n+1)} \quad \text{and} \quad b = f(\mathbb{Y}) \in \mathbb{R}^{m+1}.$$

The objective function becomes

$$\phi(\alpha_0, \alpha) = \phi(a) = \|Ma - b\|^2 = a^\top M^\top Ma - 2a^\top M^\top b + b^\top b.$$

As a convex quadratic, we can solve this problem by seeking a point where the gradient  $\nabla\phi(a)$  is the  $\mathbf{0}$  vector, thus

$$\nabla\phi(a) = 2M^\top Ma - 2M^\top b = \mathbf{0} \quad \text{implies} \quad M^\top Ma = M^\top b.$$

If  $\text{rank}(M) = n+1$ , then the square matrix  $M^\top M$  is invertible, and therefore, the unique value of  $a$  for which the gradient of  $\phi$  is null is

$$a = (M^\top M)^{-1} M^\top b.$$

The matrix  $M^\dagger := (M^\top M)^{-1} M^\top$  is called the *Moore-Penrose pseudo-inverse* of  $M$ . The Moore-Penrose pseudo-inverse arises naturally in a number of linear algebra applications, including (as we have just seen) least-squares linear regression. Notice that if  $M$  was invertible, then the Moore-Penrose pseudo-inverse would simplify to  $(M^\top M)^{-1} M^\top = M^{-1} (M^\top)^{-1} M^\top = M^{-1}$ .

Applying these results to our linear regression Problem (9.8) yields

$$(9.9) \quad \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = \left( \begin{bmatrix} \mathbf{1}^\top \\ Y \end{bmatrix} \begin{bmatrix} \mathbf{1} & Y^\top \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{1}^\top \\ Y \end{bmatrix} f(\mathbb{Y}),$$

because the assumption that  $\mathbb{Y}$  is poised for linear regression implies that the rank of the matrix  $[\mathbf{1} \ Y^\top]$  is  $n+1$ .

Similar to the simplex gradient, linear regression models result in gradient approximations with well-behaved error bounds.

**THEOREM 9.8 (Linear Regression Is Fully Linear).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$ ,  $m \geq n$ , be poised for linear regression with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}) \leq \bar{\Delta}$ . Define

$$\hat{L} := \frac{1}{\Delta} [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^m - y^0], \quad \text{and} \quad \hat{R} := \begin{bmatrix} 1 & 0 \\ \mathbf{1} & \hat{L}^\top \end{bmatrix}.$$

Let  $(\alpha_0, \alpha)$  be the solution to Equation (9.9), then

$$\begin{aligned} |f(y) - \alpha_0 - \alpha^\top y| &\leq K \left( 3 + \sqrt{m} \|\hat{R}^\dagger\| \right) \Delta^2 && \text{for all } y \in B_\Delta(x) \\ \text{and } \|\nabla f(y) - \alpha\| &\leq K \left( 1 + \frac{1}{2} \sqrt{m} \|\hat{R}^\dagger\| \right) \Delta && \text{for all } y \in B_\Delta(x). \end{aligned}$$

The proof is more difficult than that of Theorem 9.5, due to Proposition 9.2 being replaced by Equation (9.9), and the fact that we cannot use the simplification that  $L_Y(y^i) = f(y^i)$ . Details are explored in Exercises 9.17 and 9.18.

Notice that the bound in Theorem 9.8 involves the quantity  $\sqrt{m}$  instead of  $\sqrt{n}$ . As the number of regression points increase, this value increases. This is not to say that more points result in a worse approximate gradient, instead it is a result of more points creating a less accurate error bound.

## 9.5. Quadratic Models

Our final two modelling approaches are fairly simple in design, but more complex in analysis. As such, we present the approaches, but omit details of the error analysis.

Our next approach is to replace the linear interpolation model with a quadratic interpolation model. In this setting, we seek  $\alpha_0 \in \mathbb{R}$ ,  $\alpha \in \mathbb{R}^n$ , and  $H \in \mathbb{R}^{n \times n}$  to create

$$Q_Y(x) = \alpha_0 + \alpha^\top x + \frac{1}{2}x^\top Hx$$

such that  $Q_Y(y^i) = f(y^i)$  for all  $y^i \in \mathbb{Y} = \{y^0, y^1, \dots, y^m\}$ .

Without loss of generality, it make sense to enforce  $H$  symmetric (see Exercise 9.12). Under this assumption, the system of equations has  $1 + n + n(n+1)/2 = \frac{1}{2}(n+1)(n+2)$  unknowns. That is, 1 unknown for  $\alpha_0$ ,  $n$  unknowns for  $\alpha$ , and  $n(n+1)/2$  for

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,n} \\ h_{2,1} & h_{2,2} & \dots & h_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n,1} & h_{n,2} & \dots & h_{n,n} \end{bmatrix} \quad \text{with} \quad h_{i,j} = h_{j,i}.$$

Thus, we will need to evaluate  $f$  at  $\frac{1}{2}(n+1)(n+2)$  points poised for quadratic interpolation.

**DEFINITION 9.8 (Poised for Quadratic Interpolation).**

The set  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  with  $m = \frac{1}{2}(n+1)(n+2) - 1$ , is poised for quadratic interpolation if the system

$$\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top Hy^i = 0, \quad i = 0, 1, 2, \dots, m$$

has a unique (trivial) solution for  $\alpha_0, \alpha$  and  $H = H^\top$ .

If the solution to the system of equations in Definition 9.8 is unique, then any similar system but with different right-hand-side values will also have a unique solution.

**DEFINITION 9.9 (Quadratic Interpolation Function).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  with  $m = \frac{1}{2}(n+1)(n+2) - 1$ , be poised for quadratic interpolation. Then the quadratic interpolation function of  $f$  over  $\mathbb{Y}$  is

$$Q_Y(x) := \alpha_0 + \alpha^\top x + \frac{1}{2}x^\top Hx$$

where  $(\alpha_0, \alpha, H = H^\top)$  is the unique solution to

$$\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top Hy^i = f(y^i), \quad i = 0, 1, 2, \dots, m.$$

Poised for quadratic interpolation is trickier than poised for linear interpolation. In order for the linear system of equations  $\alpha_0 + \alpha^\top y^i = f(y^i)$ ,  $i \in \{0, 1, \dots, n\}$ , to have a unique solution, the points cannot be collinear, i.e., the points cannot all lie on the level surface of a single nontrivial linear function. In order for the linear system of equations  $\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top H y^i = f(y^i)$ ,  $i \in \{0, 1, \dots, \frac{1}{2}(n+1)(n+2)-1\}$ , to have a unique solution, the points cannot all lie on the level surface of a single nontrivial quadratic function (see Figure 9.2).

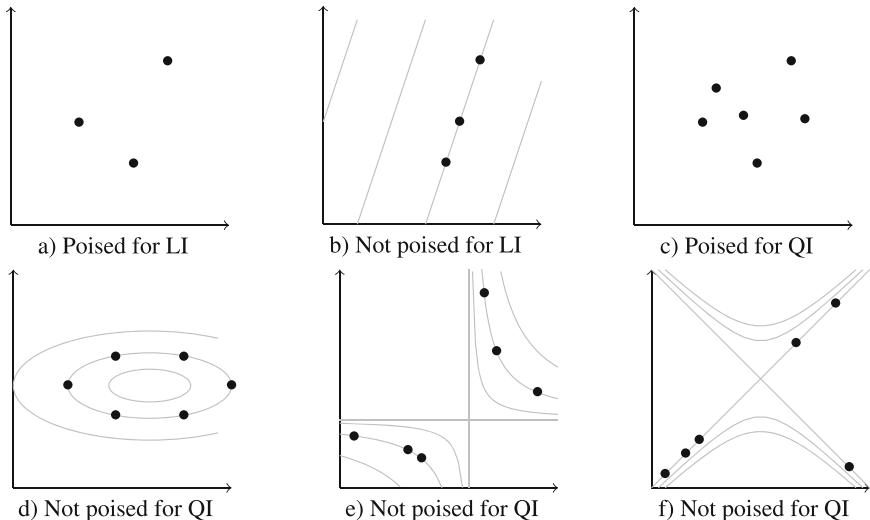


FIGURE 9.2. A set  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  is not poised for linear interpolation (LI) if and only if all the points lie on the level surface of a single nontrivial linear function; similarly, a set  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\}$ ,  $m = \frac{1}{2}(n+1)(n+2) - 1$  is not poised for quadratic interpolation (QI) if all the points lie on the level surface of a single nontrivial quadratic function

Like previous techniques, quadratic interpolation creates a class of fully linear models. In fact, quadratic interpolation results in error bounds that are stronger than fully linear. The result is what is called a class of *fully quadratic models*. At this point the exact error terms become quite technical to construct, so we satisfy ourselves with a simplified version of the error bound theorem.

**THEOREM 9.9 (Quadratic Interpolation Is Fully Quadratic).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{2+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  be poised for quadratic interpolation with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}) \leq \bar{\Delta}$ . Let  $Q_Y$  be the quadratic interpolation function of  $f$  over  $\mathbb{Y}$ . Then, there exists  $\kappa_f$ ,  $\kappa_g$ , and  $\kappa_H$  (based on  $K$ ,  $m$ , and on the “geometry of the interpolation set”) such that

$$\begin{aligned} |f(y) - Q_Y(y)| &\leq \kappa_f \Delta^3 && \text{for all } y \in B_{\Delta}(x), \\ \|\nabla f(y) - \nabla Q_Y(y)\| &\leq \kappa_g \Delta^2 && \text{for all } y \in B_{\Delta}(x) \\ \text{and } \|\nabla^2 f(y) - \nabla^2 Q_Y(y)\| &\leq \kappa_H \Delta && \text{for all } y \in B_{\Delta}(x). \end{aligned}$$

Quadratic interpolation models provide several interesting features. First, the error bounds are tighter than linear interpolation, as for  $\Delta = \overline{\text{diam}}(\mathbb{Y}) < 1$  we have  $\Delta^2 < \Delta$ . Moreover, the quadratic interpolation function  $Q_Y$  provides both an approximate gradient and an approximate Hessian. As a result,  $Q_Y$  lends itself to more advanced model-based descent algorithms, such as Newton-like methods where we use  $d = -(\nabla^2 Q_Y)^{-1} \nabla Q_Y(y^0)$  as the descent direction. These methods typically converge much faster in terms of iterations used. However, this statement should be interpreted carefully. A linear interpolation model uses approximately  $n + 1$  function evaluations per iteration, while quadratic interpolation requires approximately  $\frac{1}{2}(n + 1)(n + 2)$  function evaluations per iteration. As a result, in terms of function evaluations, a model-based method using linear interpolation can take  $(n + 2)/2$  iterations for every single iteration of a method using quadratic interpolation.

To find a better balance between the number of function calls used, and the ability of a model to provide curvature information, we turn to minimum Frobenius norm models. Minimum Frobenius norm models use between  $n + 1$  and  $(n + 1)(n + 2)/2$  points. The goal is to create an approximate gradient that is more accurate than linear interpolation, along with an approximate Hessian that is less accurate than quadratic interpolation. It is hoped that this balance can lead to DFO algorithms with strong convergence rates in terms of number of iterations, without the need for excessive function calls per iteration.

Suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\}$  where  $n + 1 < m + 1 < (n + 1)(n + 2)/2$ . Instead of thinking of this as “too many” points for linear interpolation, and using linear regression, we think of this as “too few” points for quadratic interpolation. As such, the solution  $\alpha_0, \alpha, H = H^\top$  to

$$\alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top H y^i = f(y^i) \quad \text{for } i = 0, 1, 2, \dots, m$$

is under determined (i.e., there are more variables than equations). So, if there is a solution, then the solution is not unique. In order to select a single solution from the set of all possible solutions, we aim to construct a model for which the norm of the matrix  $H$  is small to dampen the oscillatory behaviour of polynomial interpolation. As such, we define the minimum Frobenius norm

model through the solution to the following quadratic optimization problem

$$(9.10) \quad \begin{aligned} \min_{\alpha_0, \alpha, H} \quad & \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n h_{i,j}^2 \\ & \alpha_0 + \alpha^\top y^i + \frac{1}{2}(y^i)^\top H y^i = f(y^i) \quad \text{for } i = 0, 1, 2, \dots, m \\ & H = H^\top. \end{aligned}$$

The name minimum Frobenius norm model comes from the fact that the Frobenius norm of the matrix  $H$  is precisely  $\|H\| = (\sum_{i=1}^n \sum_{j=i}^n h_{i,j}^2)^{\frac{1}{2}}$  (see Section 2.1).

**DEFINITION 9.10 (Poised for Minimum Frobenius Norm Modelling).**

*The set  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  with  $n < m < \frac{1}{2}(n+1)(n+2)-1$ , is poised for minimum Frobenius norm modelling if problem (9.10) has a unique solution  $(\alpha_0, \alpha, H)$ .*

**DEFINITION 9.11 (Minimum Frobenius Norm Model Function).**

*Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$ , and  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$ ,  $n < m < \frac{1}{2}(n+1)(n+2)-1$ , be poised for minimum Frobenius norm modelling. Then the minimum Frobenius norm model function of  $f$  over  $\mathbb{Y}$  is*

$$M_Y(x) := \alpha_0 + \alpha^\top x + \frac{1}{2}x^\top Hx$$

*where  $(\alpha_0, \alpha, H)$  is the unique solution to Equation (9.10).*

The error bounds for minimum Frobenius norm models are weaker than quadratic interpolation, but as strong as linear interpolation models.

**THEOREM 9.10 (Minimum Frobenius Norm Modelling Is Fully Linear).**

*Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subset \mathbb{R}^n$  be poised for minimum Frobenius norm modelling with  $y^0 = x$  and  $\Delta = \text{diam}(\mathbb{Y}) \leq \bar{\Delta}$ . Let  $M_Y$  be the minimum Frobenius norm model function of  $f$  over  $\mathbb{Y}$ . Then, there exists  $\kappa_f$  and  $\kappa_g$  (based on  $K$ ,  $m$ , and the “geometry of the interpolation set”) such that*

$$\begin{aligned} \|f(y) - M_Y(y)\| &\leq \kappa_f \Delta^2 && \text{for all } y \in B_\Delta(x) \\ \text{and } \|\nabla f(y) - \nabla M_Y(y)\| &\leq \kappa_g \Delta && \text{for all } y \in B_\Delta(x). \end{aligned}$$

Quadratic interpolation with minimum Frobenius norm generalises linear interpolation of Section 9.2. Indeed, if the interpolation points are the  $n+1$  vertices of a simplex in  $\mathbb{R}^n$ , then the optimal solution will be to choose the matrix  $H$  to be null, resulting in  $M_Y = L_Y$ .

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 9.1.** Suppose  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$ . Prove that the first-order Taylor approximation at  $x$  will create a fully linear model of  $f$  at  $x$

**EXERCISE 9.2.** Prove Proposition 9.2.

[Hint: Examine the algebra Proposition 9.1 applied to Equation (9.3).]

**EXERCISE 9.3.** Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\} \subset \mathbb{R}^n$  be poised for linear interpolation. Given  $\gamma > 0$ , define  $\mathbb{Y}_\gamma := \{y_\gamma^0, y_\gamma^1, \dots, y_\gamma^n\}$  where  $y_\gamma^i = y^0 + \gamma(y^i - y^0)$ , and let

$$\begin{aligned}\Delta &:= \overline{\text{diam}}(\mathbb{Y}), & L &:= [y^1 - y^0 \ y^2 - y^0 \ \dots \ y^n - y^0], \\ \Delta_\gamma &:= \overline{\text{diam}}(\mathbb{Y}_\gamma), & L_\gamma &:= [y_\gamma^1 - y_\gamma^0 \ y_\gamma^2 - y_\gamma^0 \ \dots \ y_\gamma^n - y_\gamma^0].\end{aligned}$$

Prove that

$$\frac{1}{\Delta}L = \frac{1}{\Delta_\gamma}L_\gamma, \quad \text{and in particular} \quad \left\| \left( \frac{1}{\Delta}L \right)^{-\top} \right\| = \left\| \left( \frac{1}{\Delta_\gamma}L_\gamma \right)^{-\top} \right\|.$$

**EXERCISE 9.4.** Let  $f(x) = 3 + x_1 + 4x_2 + x_3 + 5x_4 + 9x_5$ . Let  $\mathbb{Y} = \{\mathbf{0}, e_1, e_2, \dots, e_5\}$ , where  $\mathbf{0}$  is the vector of all zeros, and  $e_i$  is the  $i$ -th coordinate vector in  $\mathbb{R}^5$ .

- a) Compute  $\nabla_S f(\mathbb{Y})$ .
- b) Compare  $\nabla_S f(\mathbb{Y})$  to  $\nabla f(x)$ . Explain the result.

**EXERCISE 9.5.** Let  $\tilde{f}_\Delta$  be fully linear models of  $f \in \mathcal{C}^{1+}$  with constants  $\kappa_f > 0$  and  $\kappa_g > 0$ .

- a) Let  $\lambda$  be a fixed scalar in  $\mathbb{R}$ . Are  $\lambda\tilde{f}_\Delta$  fully linear models of  $\lambda f$ ? If so, then give the constants.
- b) Are  $(\tilde{f}_\Delta)^2$  fully linear models of  $(f)^2$ ? If so, then give the constants.

**EXERCISE 9.6.** Consider  $f(x) = e^x$ .

- a) Show that given any point  $x \in \mathbb{R}$  and any  $\bar{\Delta} > 0$ , there exists a  $K$  such that  $f \in \mathcal{C}^{1+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ .
- b) Show that  $f \notin \mathcal{C}^{1+}$  with constant  $K$  for any  $K$ .
- c) Explain how this relates to the importance of  $\bar{\Delta}$  in Theorems 9.5, 9.8, 9.9, and 9.10.

**EXERCISE 9.7.** Suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  is poised for linear regression.

- a) Show that the simplex gradient  $\nabla_S f(\mathbb{Y})$  is independent of the order of the vectors in  $\mathbb{Y}$ .
- b) Prove  $\mathbb{Y}$  is poised for linear interpolation, and that the approximate gradient resulting from linear regression is the simplex gradient.

**EXERCISE 9.8.** Consider  $f(x) = x^3$  at the point  $\bar{x} = 0.5$ .

- Use  $\mathbb{Y} = \{0, 0.25, 0.5, 0.75, 1.0\}$  and least squares regression to find an approximate gradient at  $\bar{x}$ .
- Use  $\mathbb{Y} = \{0, 0.5, 1.0\}$  and least squares regression to find an approximate gradient at  $\bar{x}$ .
- Use  $\mathbb{Y} = \{0.25, 0.5, 0.75\}$  and least squares regression to find an approximate gradient at  $\bar{x}$ .
- Compare your answers above to the true gradient at  $\bar{x}$ .

**EXERCISE 9.9.**  $\square$  Consider  $f(x) = e^{x+x^3} - 2x$  at the point  $\bar{x} = 0.1$ .

- Select 10 random points in  $[-0.9, 1.1]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Select 100 random points in  $[-0.9, 1.1]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Select 1000 random points in  $[-0.9, 1.1]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Repeat a), b), and c) 10 times each and then compare your answers to the true gradient at  $\bar{x}$ .

**EXERCISE 9.10.**  $\square$  Consider  $f(x) = \sin(x^2) + \sqrt{x}$  at the point  $\bar{x} = 2$ .

- Select 11 evenly spaced points in  $[1, 3]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Select 101 evenly spaced points in  $[1, 3]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Select 1001 evenly spaced points in  $[1, 3]$  and use least squares regression to find an approximate gradient at  $\bar{x}$ .
- Compare your answers to the true gradient at  $\bar{x}$ .

**EXERCISE 9.11.** + Let  $f \in \mathcal{C}^{m+}$  with constant  $K$  and suppose  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  is poised for linear interpolation. Suppose that  $\tilde{g}_\Delta$  is an approximate gradient such that there exists a scalar  $\kappa > 0$  for which

$$\|(\nabla f(y^0) - \tilde{g}_\Delta)^\top (y^i - y^0)\| \leq \kappa \Delta^m \quad \text{for all } y^i \in B_\Delta(y^0),$$

Prove that there exists a scalar  $\bar{\kappa} > 0$  based on  $\|\hat{L}^{-1}\|$  such that

$$\|\nabla f(y^0) - \tilde{g}_\Delta\| \leq \bar{\kappa} \Delta^{m-1}.$$

**EXERCISE 9.12.** Consider the quadratic function  $Q(x) = \alpha_0 + \alpha^\top x + x^\top Mx$  where  $\alpha_0 \in \mathbb{R}$ ,  $\alpha \in \mathbb{R}^n$ , and  $M \in \mathbb{R}^{n \times n}$ . Find a symmetric matrix  $H \in \mathbb{R}^{n \times n}$  such that  $Q(x) = \alpha_0 + \alpha^\top x + x^\top Hx$  for every  $x \in \mathbb{R}^n$ .

**EXERCISE 9.13.** Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^n\}$  be  $n+1$  distinct points in  $\mathbb{R}^n$ .

- Suppose there exists a nontrivial linear function  $\beta_0 + \beta^\top x$  ( $\beta_0 \in \mathbb{R}$ ,  $\beta \in \mathbb{R}^n$ , at least one value  $\beta_i \neq 0$ ) such that  $\beta_0 + \beta^\top y^i = 0$  for each  $i \in \{0, 1, \dots, m\}$ . Prove that  $\mathbb{Y}$  is not poised for linear interpolation.
- Suppose  $\mathbb{Y}$  is not poised for linear interpolation. Prove that there exists a nontrivial linear function  $\beta_0 + \beta^\top x$  ( $\beta_0 \in \mathbb{R}$ ,  $\beta \in \mathbb{R}^n$ , at least one value  $\beta_i \neq 0$ ) such that  $\beta_0 + \beta^\top y^i = 0$  for each  $i \in \{0, 1, \dots, m\}$ .

[Parts a) and b) combined prove that  $\mathbb{Y}$  is not poised for linear interpolation if and only if all the points lie on the level surface of a single nontrivial linear function.]

**EXERCISE 9.14.** + Let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\}$ , with  $m = \frac{1}{2}(n+1)(n+2) - 1$ , be distinct points in  $\mathbb{R}^n$ . Prove that  $\mathbb{Y}$  is not poised for quadratic interpolation if and only if all the points lie on the level surface of a single nontrivial quadratic function.

**EXERCISE 9.15.** Let

$$\mathbb{Y} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right\}.$$

- a) Prove that  $\mathbb{Y}$  is **not** poised for quadratic interpolation.
- b) Sketch  $\mathbb{Y}$  in  $\mathbb{R}^2$ . Use the sketch to explain why  $\mathbb{Y}$  is not poised for quadratic interpolation.

**EXERCISE 9.16.** Let  $f(x) = (x_1)^2 + 3x_2$  and

$$\mathbb{Y} = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right\}.$$

- a) Find the quadratic interpolation model of  $f$  over  $\mathbb{Y}$ .
- b) Simplify your answer in part a), and explain the result.

**EXERCISE 9.17.** + Let  $f \in \mathcal{C}^{1+}$  with constant  $K$ , and let  $\mathbb{Y} = \{y^0, y^1, \dots, y^m\} \subseteq \mathbb{R}^n$ ,  $m \geq n$ , be poised for linear regression. Define  $\hat{f}(z) = f(z+y^0)$  and  $\hat{\mathbb{Y}} = \{y^0 - y^0, y^1 - y^0, \dots, y^m - y^0\}$ . Let  $L_Y(x) = \alpha_0 + \alpha^\top x$  be the linear regression function of  $f$  over  $\mathbb{Y}$ . Let  $\hat{L}_Y(x) = \hat{\alpha}_0 + \hat{\alpha}^\top x$  be the linear regression function of  $\hat{f}$  over  $\hat{\mathbb{Y}}$ .

- a) Prove that  $\hat{\mathbb{Y}}$  is poised for linear regression.
- b) Prove that  $\alpha = \hat{\alpha}$ .
- c) Prove that  $\alpha_0 - \hat{\alpha}_0 = -\alpha^\top y^0$ .

**EXERCISE 9.18.** + In this exercise we develop the proof of Theorem 9.8 for the case when  $y^0 = 0$ . This can be combined with Exercise 9.17 to generate the complete proof.

Let  $f \in \mathcal{C}^{1+}$  with constant  $K$ , and let  $\mathbb{Y} = \{0, y^1, \dots, y^m\} \subseteq \mathbb{R}^n$ ,  $m \geq n$ , be poised for linear regression. Define  $\Delta$ ,  $\hat{L}$ , and  $\hat{R}$  as in Theorem 9.8.

- a) Use Lemma 9.4 to prove that

$$\left\| \hat{R} \begin{bmatrix} f(y^0) \\ \nabla f(y^0) \end{bmatrix} - f(\mathbb{Y}) \right\| \leq \frac{1}{2} \sqrt{m} K \Delta^2, \text{ where } f(\mathbb{Y}) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^m) \end{bmatrix}.$$

- b) Prove

$$\begin{bmatrix} f(y^0) \\ \nabla f(y^0) \end{bmatrix} - \hat{R}^\dagger f(\mathbb{Y}) = \hat{R}^\dagger r \quad \text{where} \quad r = \hat{R} \begin{bmatrix} f(y^0) \\ \nabla f(y^0) \end{bmatrix} - f(\mathbb{Y}),$$

and where  $\hat{R}^\dagger$  is the Moore-Penrose pseudo-inverse of  $\hat{R}$ .

- c) Use Equation (9.9) and  $|r| \leq \frac{1}{2}\sqrt{m}K\Delta^2$ , from part a), to derive the error bounds in Theorem 9.8.  
 [Hint: for part c), first derive error bounds at  $y^0$ , then derive the final results.]

□

**Chapter 9 Project: The Centred Simplex Gradient.** A powerful trick to improving accuracy in an approximate gradient is the centred simplex gradient. It relies on the well-known results in numerical analysis that for a function  $f \in \mathcal{C}^1$  of a single variable, its derivative may be approximated by the forward and the centred finite differences

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \quad \text{and} \quad f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2).$$

The centred difference formulae is more precise than the forward difference.

The centred simplex gradient begins with a set poised for linear interpolation  $\mathbb{Y}^+ = \{y^0, y^1, \dots, y^n\}$ . From  $\mathbb{Y}^+$ , we define  $d^i = y^i - y^0$  for  $i = 1, 2, \dots, n$  and introduce a second set of  $n + 1$  points denoted  $\mathbb{Y}^- = \{y^0, y^0 - d^1, y^0 - d^2, \dots, y^0 - d^n\}$ . This second set is also poised for linear interpolation (see part a) below). Notice that  $\mathbb{Y}^+$  may also be written as  $\mathbb{Y}^+ = \{y^0, y^0 + d^1, y^0 + d^2, \dots, y^0 + d^n\}$ , so these two sets are reflections of each other through the point  $y^0$ . As  $\mathbb{Y}^-$  and  $\mathbb{Y}^+$  are both poised for linear interpolation, we can create the two simplex gradients  $\nabla_S f(\mathbb{Y}^-)$  and  $\nabla_S f(\mathbb{Y}^+)$ . The centred simplex gradient is the average of these two simplex gradients.

**DEFINITION 9.12 (Centred Simplex Gradient).**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and let  $\mathbb{Y}^+ = \{y^0, y^1, \dots, y^n\} \subseteq \mathbb{R}^n$  be poised for linear interpolation. Define  $d^i = y^i - y^0$  for  $i = 1, 2, \dots, n$  and  $\mathbb{Y}^- = \{y^0, y^0 - d^1, y^0 - d^2, \dots, y^0 - d^n\}$  and set  $\mathbb{Y} = \mathbb{Y}^+ \cup \mathbb{Y}^-$ . The Centred Simplex Gradient of  $f$  over  $\mathbb{Y}$ , denoted  $\nabla_{CS} f(\mathbb{Y})$ , is given by

$$\nabla_{CS} f(\mathbb{Y}) := \frac{1}{2} (\nabla_S f(\mathbb{Y}^-) + \nabla_S f(\mathbb{Y}^+)).$$

Like the simplex gradient, the centred simplex gradient can be calculated directly.

**PROPOSITION 9.11.**

Let  $f : \mathbb{R}^n \mapsto \mathbb{R}$  and let  $\mathbb{Y}^+ = \{y^0, y^1, \dots, y^n\} \subseteq \mathbb{R}^n$  be poised for linear interpolation. Define  $d^i := y^i - y^0$  for  $i = 1, 2, \dots, n$ ,

$$L := [y^1 - y^0 \quad y^2 - y^0 \quad \dots \quad y^n - y^0] \quad \text{and}$$

$$\delta_{CS}^{f(\mathbb{Y})} := \frac{1}{2} \begin{bmatrix} f(y^1) - f(y^0 - d^1) \\ f(y^2) - f(y^0 - d^2) \\ \vdots \\ f(y^n) - f(y^0 - d^n) \end{bmatrix}.$$

Then the centred simplex gradient  $\alpha = \nabla_{CS} f(\mathbb{Y})$  is the solution of the linear system

$$(9.11) \quad L^\top \alpha = \delta_{CS}^{f(\mathbb{Y})}.$$

Notice that Proposition 9.11 allows the computation of the centred simplex gradient using just  $2n$  function evaluations, as  $f(y^0)$  is not actually required in Equation (9.11). However, note that the centred simplex is still dependent on  $y^0$ , as it provides the centring point for the approximation.

Finally, using slightly stronger conditions on  $f$  ( $f \in \mathcal{C}^{2+}$  rather than  $f \in \mathcal{C}^{1+}$ ), it can be shown that the centred simplex gradient is a good approximation of the true gradient. In fact, the error bound for the centred simplex gradient improves on the error bound for the simplex gradient, as it is dependent on  $\overline{\text{diam}}(\mathbb{Y})^2$  instead of  $\overline{\text{diam}}(\mathbb{Y})$ .

**PROPOSITION 9.12.**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{2+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y}^+$  be poised for linear interpolation with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}^+) \leq \bar{\Delta}$ . Then for all  $i = 1, 2, \dots, n$ , the centred simplex gradient  $\alpha = \nabla_{CS} f(\mathbb{Y})$  satisfies

$$\|(y^i - y^0)^\top (\nabla f(y^0) - \alpha)\| \leq \frac{K}{6} \Delta^3.$$

**THEOREM 9.13 (Centred Simplex Gradient Error Bound).**

Let  $x \in \mathbb{R}^n$  and  $f \in \mathcal{C}^{2+}$  on  $B_{\bar{\Delta}}(x)$  with constant  $K$ . Let  $\mathbb{Y}^+$  be poised for linear interpolation with  $y^0 = x$  and  $\Delta = \overline{\text{diam}}(\mathbb{Y}^+) \leq \bar{\Delta}$ . Then the centred simplex gradient  $\alpha = \nabla_{CS} f(\mathbb{Y})$  satisfies

$$(9.12) \quad \|\nabla f(y^0) - \alpha\| \leq \frac{K}{6} \sqrt{n} \|\hat{L}^{-1}\| \Delta^2.$$

where  $\hat{L}$  is the matrix from Equation (9.4). Furthermore,

$$(9.13) \quad \|\nabla f(y) - \alpha\| \leq \left( K + \frac{K}{6} \sqrt{n} \|\hat{L}^{-1}\| \right) \Delta^2, \quad \text{for all } y \in B_\Delta(y^0).$$

The goal of this project is to complete the missing details and prove the statements above.

- a) Let  $\mathbb{Y}^+ = \{y^0, y^1, \dots, y^n\}$  be poised for linear interpolation. Show that

$$\mathbb{Y}^- = \{y^0, y^0 - d^1, y^0 - d^2, \dots, y^0 - d^n\}$$

where  $d^i = y^i - y^0$  for  $i = 1, 2, \dots, n$ , is also poised for linear interpolation.

- b) Prove Proposition 9.11.  
c) Prove Proposition 9.12.  
d) Prove Theorem 9.13.

[Hint: use Exercise 9.11]

## *Model-Based Descent*

Model-based methods in DFO proceed from the idea that if it is possible to build a “good” model of the true objective function, then information from the model can be used to guide the optimization. In Chapter 9, we studied several methods for constructing model functions using only objective function evaluations. We also defined *fully linear*, as a term to mean that the optimizer has access to an *accuracy parameter*  $\Delta$  that can be used to drive the error in the model to 0 in a predictable way. We now turn our attention to how to employ model functions in unconstrained DFO algorithms.

Most model-based DFO methods fall into one of two categories. First, methods where the model is used to help find a *descent direction* of the objective function, whereupon a line-search is used to improve the incumbent solution. Second, methods where the model is minimised subject to some form of *trust region constraint* (or penalty) to determine a *trial point*, whereas the true objective function is evaluated to determine if the incumbent solution should be updated. In both cases, we must take careful control of the accuracy parameter  $\Delta$ , driving it to 0 as the algorithm converges towards a prospective minimiser. In order to ensure convergence, we must also ensure that the incumbent solution is only updated when sufficient decrease in the objective function value occurs.

In Chapter 11, we will study a model-based trust-region (MBTR) algorithm for DFO, and in this chapter we focus on model-based descent (MBD) algorithm. Similar to the GPS and MADS algorithms, the MBD and MBTR algorithms include methods to allow meta-heuristic methods to be embedded within the algorithm. Unlike the GPS and MADS algorithms, these methods are not placed at the beginning of any iteration in the form of a search step, but are placed at the end of an iterate in the form of a flexible update step.

## 10.1. Controllable Accuracy

In Chapter 9, model “goodness” was defined in terms of *fully linear* models. Fully linear implies that the optimizer has control over a parameter that controls the model error in both function value and gradient value. The MBD algorithm discussed in this chapter uses the first-order information from the model (i.e., the approximate gradients) to determine descent directions, and then performs a line-search to seek improvement in that direction. As such, the MBD algorithm does not need the model to have “good” function values, only “good” gradient approximations.

**DEFINITION 10.1** (Controllably Accurate Gradient Approximations at  $x$ ).  
*Given  $f \in \mathcal{C}^1$ ,  $x \in \mathbb{R}^n$ , and  $\bar{\Delta} > 0$  we say that  $\{\tilde{g}_\Delta(x) \in \mathbb{R}^n : \Delta \in (0, \bar{\Delta}]\}$  are controllably accurate approximations of the gradient  $\nabla f$  at  $x$  if there exists a scalar  $\kappa_g(x) > 0$  such that given any  $\Delta \in (0, \bar{\Delta}]$ ,*

$$\|\nabla f(x) - \tilde{g}_\Delta(x)\| \leq \kappa_g(x)\Delta.$$

*We say  $\tilde{g}_\Delta$  are controllably accurate approximations of the gradient  $\nabla f$  (without mention of  $x$  or  $\bar{\Delta}$ ) to mean that given any  $x \in \mathbb{R}^n$  and  $\bar{\Delta}$ , there exists a set  $\{\tilde{g}_\Delta(x) \in \mathbb{R}^n : \Delta \in (0, \bar{\Delta}]\}$  of controllably accurate gradient approximations. Finally, we say  $\tilde{g}_\Delta$  are controllably accurate approximations of the gradient  $\nabla f$  with constant  $\kappa_g$  to mean that  $\tilde{g}_\Delta$  are controllably accurate approximations of the gradient  $\nabla f$  and the scalar  $\kappa_g(x)$  can be taken as the constant  $\kappa_g$ .*

It should be immediately obvious that given a class of fully linear models, we immediately have approximate gradients with *controllable accuracy*. As such, Chapter 9 has already provided a number of methods to generate controllably accurate approximations of the gradient. This also highlights the role of  $\bar{\Delta}$  in Definition 10.1.

Like fully linear models, when assuming controllable gradient accuracy, it should be envisioned that the constant  $\kappa_g(x)$  is unknown to the optimizer/algorithm, but the parameter  $\Delta$  is under the control of the optimizer/algorithm. In practice, the algorithm will have no knowledge of what value the constant  $\kappa_g(x)$  might take, only that it is fixed. Surprisingly, this is sufficient to guarantee convergence in many situations.

It is not difficult to create a model function (parameterised by  $\Delta$ ) that provides controllably accurate approximations of the gradient, but does not come from a class of fully linear models (see Exercise 10.1). However, the two notions are actually extremely similar, as the next proposition demonstrates.

**PROPOSITION 10.1.**

Suppose  $x \in \mathbb{R}^n$ ,  $\bar{\Delta} > 0$ ,  $f \in \mathcal{C}^{1+}$  with constant  $K$  on  $B_{\bar{\Delta}}(x)$ .

Suppose the model functions  $\tilde{f}_\Delta \in \mathcal{C}^{1+}$  are such that their gradients  $\{\nabla \tilde{f}_\Delta : \Delta \in (0, \bar{\Delta}]\}$  are controllably accurate approximations of  $\nabla f$ .

If  $\tilde{f}_\Delta(x) = f(x)$  for every  $\Delta \in (0, \bar{\Delta}]$ , then  $\{\tilde{f}_\Delta\}$  is a class of fully linear models of  $f$  at  $x$ .

**PROOF.** See Exercise 10.2. □

In light of Proposition 10.1, it is clear that assuming the existence of fully linear model or the existence of controllably accurate gradient approximations is essentially the same. In this chapter, we focus on the existence of controllably accurate gradient approximations in order to emphasise that the MBD algorithm only uses the approximate gradients to generate descent directions.

## 10.2. The MBD Algorithm

Model-based methods can be broadly split into two groups, trust region methods and descent-based methods. Informally, trust region methods (covered in Chapter 11) rely on the assumption that the model is “trustworthy” within some region enclosing the algorithmic centre, and use the minimisers of the model restricted to this “trust region” to guide the next iterate. Descent-based methods (covered in this chapter) use the model’s gradient information to determine a descent direction (recall Definition 2.4) of the true function and then perform a line-search in that direction.

At the incumbent solution  $x^k$ , once that a descent direction  $d^k$  for the objective function  $f$  is found, a descent-based method performs a line-search in that direction. A line-search simply takes a step from the current incumbent solution  $x^k$  of length  $t^k$  relative to the direction  $d^k$ : the algorithm evaluates  $f(x^k + t^k d^k)$ . However, unlike the three direct-search methods (CS, GPS, and MADS), the next incumbent solution needs to satisfy more stringent conditions than a mere decrease in objective function value. Indeed, a direct-search method accepts  $x^k + t^k d^k$  as the next incumbent if  $f(x^k + t^k d^k) < f(x^k)$ , but the line-search method requires a minimal decrease on the objective function value. This is called the Armijo condition and will be described in detail in Section 10.4. Direct-search methods do not require a minimal decrease condition, but instead require that each trial point be located on the mesh.

A pseudo-code of the MBD algorithm for unconstrained optimization is next. Within the code we assume that controllably accurate gradient approximations  $\tilde{g}_\Delta$  are given for  $\nabla f$ . In order to present cleaner proofs, we shall assume that  $\Delta^k \leq \bar{\Delta}$  in all iterations of the MBD algorithm (this is permitted under Definition 10.1). Thus, the error bound of Definition 10.1 can be applied without complicated phrasing.

---

**ALGORITHM 10.1.** Model-based descent (MBD)

---

Given  $f \in \mathcal{C}^1$ , starting point  $x^0 \in \mathbb{R}^n$  and controllably accurate gradient approximations  $\tilde{g}_\Delta$  of  $\nabla f$

0. Initialise:

|  |                                   |
|--|-----------------------------------|
| $\Delta^0 \in (0, \infty)$               | initial model accuracy parameter  |
| $\mu^0 \in (0, \infty)$                  | initial target accuracy parameter |
| $\eta \in (0, 1)$                        | an Armijo parameter               |
| $\varepsilon_d \in (0, 1)$               | minimum decrease angle parameter  |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance                |
| $k \leftarrow 0$                         | iteration counter                 |

1. Model:

use  $\Delta^k$  and a finite number of points to create controllably accurate gradient approximations: Set  $\tilde{g}^k = \tilde{g}_{\Delta^k}(x^k)$

2. Model accuracy checks:

a) if  $\Delta^k < \epsilon_{\text{stop}}$  and  $\|\tilde{g}^k\| < \epsilon_{\text{stop}}$   
declare algorithm success and stop

b) if  $\Delta^k > \mu^k \|\tilde{g}^k\|$   
declare the model inaccurate  
decrease  $\Delta^{k+1} \leq \frac{1}{2}\Delta^k$ , set  $\mu^{k+1} = \mu^k, x^{k+1} = x^k,$   
 $k \leftarrow k + 1$ , go to 1

c) if  $\Delta^k \leq \mu^k \|\tilde{g}^k\|$   
declare the model accurate and proceed to 3

3. Line search:

select  $d^k \in \mathbb{R}^n$  such that  $\left(\frac{d^k}{\|d^k\|}\right)^\top \left(\frac{\tilde{g}^k}{\|\tilde{g}^k\|}\right) < -\varepsilon_d$  (descent)

perform a line-search in the direction  $d^k$  to seek  $t^k$  with

$f(x^k + t^k d^k) < f(x^k) + \eta t^k (d^k)^\top \tilde{g}^k$  (Armijo)

if  $t^k$  is found declare line-search success

otherwise declare line-search failure

4. Update:

if line-search success

let  $x^{k+1}$  be any point such that  $f(x^{k+1}) \leq f(x^k + t^k d^k)$

set  $\Delta^{k+1} = \Delta^k, \mu^{k+1} = \mu^k$

otherwise (line-search failure)

set  $x^{k+1} = x^k, \Delta^{k+1} = \Delta^k$ , and decrease  $\mu^{k+1} \leq \frac{1}{2}\mu^k$

increment  $k \leftarrow k + 1$  and go to 1

Our primary goal in this chapter is to prove the convergence of the MBD algorithm by showing that the sequence of iterates  $x^k$  satisfies  $\|\nabla f(x^k)\| \rightarrow 0$ . But first, let us make a few comments about the MBD algorithm.

### 10.3. Flexibility in the MBD Algorithm and Stopping Criteria

Like the direct search methods in Part 3, the MBD algorithm is a local optimization method, i.e., it starts at an initial point and seeks a highly accurate local minimiser. As such, a good starting point can greatly improve the effectiveness of the algorithm. The comments and discussion in Sections 3.4 and 7.2 apply directly to the MBD algorithm.

The MBD algorithm incorporates a surprising amount of flexibility in its description. First, note that the selection of the search direction  $d^k$  (in Step 3) is left somewhat vague. It should be clear that selecting a descent direction  $d^k$  is always possible. Indeed, if  $\tilde{g}^k \neq 0$ , then setting  $d^k = -\tilde{g}^k$  will always satisfy the descent condition (see Exercise 10.3). While, if  $\tilde{g}^k = 0$ , then the model check stage in Step 2b) would not have allowed the MBD algorithm to continue to Step 3. The reason the descent direction is left flexible is that this allows other descent directions to be considered. For example, a quasi-Newton descent direction ( $-H^{-1}\nabla g^k$ , where  $H$  is a positive definite matrix) could be considered, although its success will depend on the value of  $\epsilon_d$  (see Exercise 10.3).

The methods in Part 3 include a **search step**, which allows the algorithm to apply heuristics to break free of local minimisers. In the MBD algorithm, there is no “search step”, but the algorithm allows for flexibility during the update step. In particular, note that in Step 4, after a line-search success the algorithm may select any  $x^{k+1}$  such that  $f(x^{k+1}) \leq f(x^k + t^k d^k)$ . While  $x^{k+1} = x^k + t^k d^k$  is clearly a valid (and easy) option, at this stage the algorithm could alternately select an  $x^{k+1}$  from the list of points evaluated to create the approximate gradient, or apply some subroutine designed to break free of local minimisers. For example, a line-search success could trigger a few iterations of GA (or some other search heuristic). Unlike the methods in Part 3, there is no *mesh* in the MBD algorithm, so no need for the search heuristic to restrict its search to a specific set.

One of the major differences between model-based methods and the methods in Parts 2 and 3 is the stopping condition. Unlike heuristic methods and direct search methods, model-based methods have an approximate gradient to work with. As a result of this extra information, stopping conditions for unconstrained optimization can clearly be set based on the approximated gradient becoming sufficiently small. Of course, one wants the approximate gradient to be small because the true gradient is small, not because of inaccuracy in the approximations. As such, the stopping condition in Step 2a) includes a condition that the radius  $\Delta^k$  must also be small.

Recall that the accuracy parameter is assumed to satisfy  $\Delta^k \leq \bar{\Delta}$  in all iterations of the MBD algorithm, so the error bound of Definition 10.1 can be applied without complicated phrasing.

**PROPOSITION 10.2.**

Suppose  $f \in \mathcal{C}^1$  and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$ . If, at iteration  $k$ , MBD stops at Step 2a), then

$$\|\nabla f(x^k)\| \leq (\kappa_g(x^k) + 1)\epsilon_{\text{stop}},$$

where  $\kappa_g(x^k)$  is the controllable accuracy parameter of the gradient at  $x^k$ .

**PROOF.** Applying the triangle inequality to the controllable accuracy inequality yields

$$\|\nabla f(x^k)\| - \|\tilde{g}^k\| \leq \|\nabla f(x^k) - \tilde{g}^k\| = \|\nabla f(x^k) - \tilde{g}_{\Delta^k}(x^k)\| \leq \kappa_g(x^k)\Delta^k$$

where  $\tilde{g}^k = \tilde{g}_{\Delta^k}(x^k)$ . Next, since  $\|\tilde{g}^k\| < \epsilon_{\text{stop}}$  and  $\Delta^k < \epsilon_{\text{stop}}$  we have

$$\|\nabla f(x^k)\| - \epsilon_{\text{stop}} \leq \kappa_g(x^k)\epsilon_{\text{stop}},$$

which is equivalent to the desired result.  $\square$

Of course, in practice we do not know the value of  $\kappa_g(x^k)$  from Definition 10.1. Nonetheless, Proposition 10.2 provides a strong relationship between algorithm success and classical unconstrained first order optimality. Assuming  $\kappa_g(x^k)$  is reasonably small, by making  $\epsilon_{\text{stop}}$  sufficiently small, approximate first order optimality is achieved.

#### 10.4. The MBD Algorithm Has Successful Iterations

In order to prove convergence of the MBD algorithm, we must first show that any given iteration will eventually be successful. To do this, we must show that we will not get stuck in an infinite loop of inaccurate models in Step 2. Then, we must accept that a descent direction can always be found (which we discussed in Section 10.3). Finally, we must show that the *Armijo condition* in Step 3,

$$(10.1) \quad f(x^k + t^k d^k) < f(x^k) + \eta t^k (d^k)^\top \tilde{g}^k$$

can always be satisfied if the step size parameter  $t^k > 0$  is sufficiently small.

We now examine Step 2b), where the MBD algorithm checks if the model is sufficiently accurate to work with.

If the model is not sufficiently accurate, then the MBD algorithm requests greater accuracy at  $x^k$  by decreasing the accuracy parameter  $\Delta^k$ . As  $\Delta^k$  goes to 0, the approximate gradient converges to the true gradient, and one of two things should occur. Either,  $\Delta^k \rightarrow 0$  and  $\|\tilde{g}^k\| \rightarrow 0$ , in which case the stopping test in 2a) should eventually declare success; or  $\Delta^k \rightarrow 0$  and  $\|\tilde{g}^k\| \rightarrow \gamma > 0$ , in which case the model should eventually be declared accurate. This is formalised in the next proposition.

Note the role of  $\mu^k$  in Proposition 10.3. As line-search failures occur,  $\mu^k$  is driven to 0, thus demanding higher and higher model accuracy as the algorithm progresses.

**PROPOSITION 10.3.**

Suppose  $f \in \mathcal{C}^1$  and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$ . Then MBD can only declare the model inaccurate a finite number of times in a row, i.e., MBD cannot get stuck in a loop between Step 1 and Step 2b).

**PROOF.** For eventual contradiction, suppose the iteration number  $\bar{k}$  is such that

$$\Delta^k > \mu^k \|\tilde{g}^k\| \quad \text{for all } k \geq \bar{k}.$$

Note that under these conditions  $\mu^k = \mu^{\bar{k}}$  and  $x^k = x^{\bar{k}}$  for all  $k \geq \bar{k}$ . Therefore, to ease notation, we set  $\bar{\mu} = \mu^{\bar{k}}$  and  $\bar{x} = x^{\bar{k}}$  and  $\bar{\kappa}_g$  be the controllable accuracy parameter for the models of  $f$  at  $\bar{x}$ . Furthermore, notice that  $\Delta^k \leq (\frac{1}{2})^{k-\bar{k}}$  for all  $k \geq \bar{k}$ , and in particular,  $\Delta^k \rightarrow 0$ . We now break the analysis into two disjoint cases.

**Case I:** If  $\nabla f(\bar{x}) = 0$ , then consider  $k$  sufficiently large such that

$$\Delta^k < \min \left\{ \frac{\epsilon_{\text{stop}}}{\bar{\kappa}_g}, \epsilon_{\text{stop}} \right\}.$$

By controllable accuracy,  $\|\nabla f(\bar{x}) - \tilde{g}^k\| \leq \bar{\kappa}_g \Delta^k$  and as  $\nabla f(\bar{x}) = 0$  and  $\Delta^k < \frac{\epsilon_{\text{stop}}}{\bar{\kappa}_g}$ , this implies

$$\|\tilde{g}^k\| < \bar{\kappa}_g \frac{\epsilon_{\text{stop}}}{\bar{\kappa}_g} = \epsilon_{\text{stop}}.$$

Furthermore, as  $\Delta^k < \epsilon_{\text{stop}}$ , the stopping test in Step 2a) succeeds and the MBD algorithm stops.

**Case II:** If  $\nabla f(\bar{x}) \neq 0$ , then the triangle inequality applied to controllable accuracy provides

$$\|\nabla f(\bar{x})\| - \|\tilde{g}^k\| \leq \|\nabla f(\bar{x}) - \tilde{g}^k\| \leq \bar{\kappa}_g \Delta^k.$$

Multiplying by  $\bar{\mu}$  and rearranging yield

$$(10.2) \quad \bar{\mu} \|\nabla f(\bar{x})\| - \bar{\mu} \bar{\kappa}_g \Delta^k \leq \bar{\mu} \|\tilde{g}^k\|.$$

Therefore, if  $k$  is large enough so that

$$(10.3) \quad \Delta^k \leq \frac{\bar{\mu}}{(1 + \bar{\mu} \bar{\kappa}_g)} \|\nabla f(\bar{x})\|,$$

then this may be rewritten as

$$\Delta^k \leq \bar{\mu} \|\nabla f(\bar{x})\| - \bar{\mu} \bar{\kappa}_g \Delta^k.$$

Combining this with Equation (10.2) implies that Step 2c) would declare the model accurate.  $\square$

Having established that Step 2 will eventually lead to Step 3, unless the algorithm terminates successfully, we now examine the line-search conditions

in Step 3. A useful fact that will be frequently used in what follows is the following equivalent formulation of the descent condition:

$$(10.4) \quad \left( \frac{d^k}{\|d^k\|} \right)^\top \left( \frac{\tilde{g}^k}{\|\tilde{g}^k\|} \right) < -\varepsilon_d \quad \Leftrightarrow \quad \varepsilon_d \|d^k\| \|\tilde{g}^k\| < -(d^k)^\top \tilde{g}^k.$$

Step 3 begins by requesting the creation of a search direction  $d^k$  satisfying Equation (10.4). Recall from Section 2.2, that if  $f \in \mathcal{C}^1$ , then the directional derivative in the direction  $d$  can be computed by the scalar product

$$f'(\bar{x}; d) = (d)^\top \nabla f(\bar{x}).$$

If this value is negative, then locally the function slopes downwards in the direction  $d$ . Therefore, if  $\nabla f(\bar{x}) \neq 0$ , then it is always possible to determine a descent direction by selecting  $d = -\nabla f(\bar{x})/\|\nabla f(\bar{x})\|$ . Indeed, if  $d = -\nabla f(\bar{x})/\|\nabla f(\bar{x})\|$ , then

$$f'(\bar{x}; d) = -\|\nabla f(\bar{x})\| < 0.$$

If  $d$  is a unit vector, then this is the most negative the directional derivative can be. As such,  $d = -\nabla f(\bar{x})/\|\nabla f(\bar{x})\|$  is commonly called the *direction of steepest descent*.

Of course, there may be times when the direction of steepest descent is not the best search direction to use. For example,  $d = -H^{-1}\nabla f(\bar{x})$ , where  $H$  is any symmetric positive definite matrix, is also a descent direction (see Exercise 10.3). If curvature information is known (or approximated), then using second order derivatives and setting  $H \approx \nabla^2 f(\bar{x})$  is referred to as a *quasi-Newton direction*, and often demonstrates better convergence.

Returning to first-order derivatives and to Equation (10.4), we notice a few differences between it and simply asking for a descent direction for the objective function  $f$ . First, and most obvious, is the MBD algorithm does not have access to the gradient  $\nabla f(x^k)$ , so instead our approximation  $\tilde{g}^k = \tilde{g}_{\Delta^k}(x^k)$  is used. Second, instead of simply asking for  $f'(x^k; d^k)$  to be negative, the MBD algorithm requires that the approximation of  $f'(x^k; d^k)$  be less than the strictly negative value  $-\varepsilon_d \|d\| \|\tilde{g}^k\|$  where  $\varepsilon_d$  is a fixed parameter. As  $\varepsilon_d \in (0, 1)$ , this requirement is always achievable by using the steepest descent direction for the model

$$d^k = -\tilde{g}^k \quad \text{or} \quad d^k = -\frac{\tilde{g}^k}{\|\tilde{g}^k\|}.$$

The purpose of  $\varepsilon_d$  is to provide a safety zone, so that even when  $\tilde{g}^k$  contains inaccuracy, we can still guarantee that  $d^k$  is also a descent direction for  $f$ . The next proposition gives a formal proof. Its statement requires the internal algorithmic parameters  $\mu^k$  (the accuracy parameter) and  $\varepsilon_d$  (the descent direction parameter).

**PROPOSITION 10.4.**

Suppose  $f \in \mathcal{C}^1$  and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$ . Applying MBD, suppose  $\nabla f(x^k) \neq 0$  and

$$\mu^k < \frac{\varepsilon_d}{2\kappa_g(x^k)} \quad \text{and} \quad \Delta^k < \mu^k \|\tilde{g}^k\|,$$

where  $\kappa_g(x^k)$  is the controllable accuracy parameter for the models of  $f$  at  $x^k$ . Then

$$\left( \frac{d^k}{\|d^k\|} \right)^\top \left( \frac{\tilde{g}^k}{\|\tilde{g}^k\|} \right) < -\varepsilon_d \quad \Rightarrow \quad (d^k)^\top \nabla f(x^k) < 0.$$

In particular, if  $\mu^k$  is sufficiently small, then Step 3 implies  $d^k$  is a descent direction for  $f$  at  $x^k$ .

**PROOF.** The condition  $\Delta^k < \mu^k \|\tilde{g}^k\|$  implies that  $\|\tilde{g}^k\| \neq 0$ . Applying controllably accuracy and the upper bounds on  $\mu^k$  and  $\Delta^k$  ensures

$$\begin{aligned} (10.5) \quad \|\nabla f(x^k) - \tilde{g}^k\| &= \|\nabla f(x^k) - \tilde{g}_{\Delta^k}(x^k)\| \\ &\leq \kappa_g(x^k) \Delta^k \\ &< \frac{\varepsilon_d}{2\mu^k} \mu^k \|\tilde{g}^k\| \\ &= \frac{\varepsilon_d}{2} \|\tilde{g}^k\|. \end{aligned}$$

Furthermore, we have

$$\begin{aligned} (d^k)^\top \nabla f(x^k) &= (d^k)^\top (\nabla f(x^k) - \tilde{g}^k) + (d^k)^\top \tilde{g}^k \\ &\leq \|d^k\| \|\nabla f(x^k) - \tilde{g}^k\| + (d^k)^\top \tilde{g}^k \\ &< \|d^k\| \|\nabla f(x^k) - \tilde{g}^k\| - \varepsilon_d \|d^k\| \|\tilde{g}^k\| \quad \text{by (10.4)} \\ &< \|d^k\| \frac{\varepsilon_d}{2} \|\tilde{g}^k\| - \varepsilon_d \|d^k\| \|\tilde{g}^k\| \quad \text{by (10.5)} \\ &= -\frac{\varepsilon_d}{2} \|d^k\| \|\tilde{g}^k\| \\ &< 0. \end{aligned}$$

The theorem's final statement takes into account that  $\Delta^k < \mu^k \|\tilde{g}^k\|$  is established in Step 2 of the MBD algorithm.  $\square$

Step 3 next applies a line-search to seek a step-length  $t^k$  satisfying the Armijo condition

$$f(x^k + t^k d^k) < f(x^k) + \eta t^k (d^k)^\top \tilde{g}^k.$$

To understand this equation, we begin by examining the linear function

$$L(t) = f(x^k) + t(d^k)^\top \nabla f(x^k).$$

This function is the linearisation (i.e., the tangent line, or the first-order Taylor approximation) of  $f$  at  $x^k$  in the direction  $d^k$ :  $L(t) \approx f(x^k + td^k)$ . As

such,  $L(t)$  provides a predicted level of decrease that the MBD algorithm can expect in the direction  $d^k$ . Of course, this decrease is unlikely to be perfectly obtained, so the *Armijo condition* introduces a small tilt to the function to create

$$L_\eta(t) = f(x^k) + \eta t(d^k)^\top \nabla f(x^k).$$

Notice that  $(d^k)^\top \nabla f(x^k) < 0$  and  $\eta \in (0, 1)$  ensure that  $L_\eta(t) > L(t)$  for  $t > 0$ . Thus, seeking a step-length  $t$  such that  $L_\eta(t) > f(x^k + td^k)$  is easier to obtain than seeking a step-length  $t$  such that  $L(t) > f(x^k + td^k)$ . Indeed, the second of these may never occur, but “Armijo’s Theorem” states that if  $f \in \mathcal{C}^1$ , then there is an open interval of step lengths such that  $L_\eta(t) > f(x^k + td^k)$ . Figure 10.1 illustrates the objective function  $f(x^k + td^k)$  evaluated by taking a step of length  $t$  in the direction  $d^k$  from  $x^k$ . The linear approximations  $L(t)$  and  $L_\eta(t)$  (the dotted line) also appear in light grey.

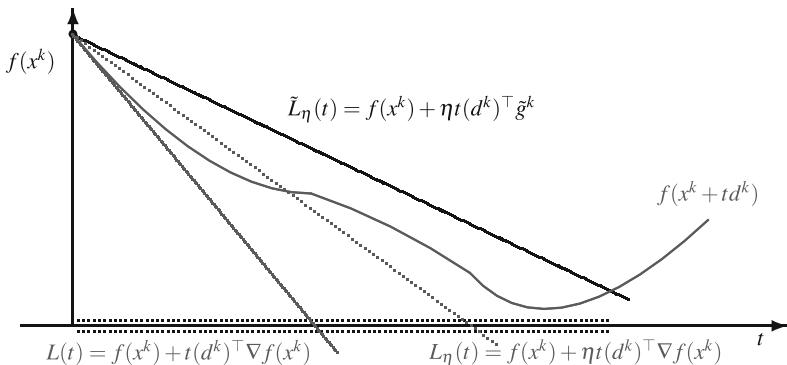


FIGURE 10.1. A visualisation of the Armijo condition of MBD

As usual, we do not have access to  $\nabla f(x^k)$ , so we replace it with the controllably accurate gradient approximation  $\tilde{g}^k = \tilde{g}_{\Delta^k}(x^k)$  and introduce:

$$\tilde{L}_\eta(t) = f(x^k) + \eta t(d^k)^\top \tilde{g}^k.$$

This linear approximation is represented by the dark line in Figure 10.1. The set of values of  $t$  that satisfy the Armijo condition is represented in the figure by the dotted interval on the horizontal axis. While this approximation of  $L_\eta$  by  $\tilde{L}_\eta$  may cause some additional error in the linearisation, Armijo’s Theorem is easily adapted to this setting.

**THEOREM 10.5 (Armijo Line-Search Condition in DFO).**

Suppose  $f \in \mathcal{C}^1$  and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$ . Applying MBD, suppose  $\nabla f(x^k) \neq 0$  and

$$\mu^k < \frac{\varepsilon_d}{2\kappa_g(x^k)} \quad \text{and} \quad \mu^k < \frac{\varepsilon_d}{\kappa_g(x^k)} \left( \frac{1-\eta}{1+\eta} \right),$$

where  $\kappa_g(x^k)$  is the controllable accuracy parameter for the models of  $f$  at  $x^k$ . Suppose  $d^k$  is selected such that

$$\Delta^k < \mu^k \|\tilde{g}^k\| \quad \text{and} \quad \left( \frac{d^k}{\|d^k\|} \right)^\top \left( \frac{\tilde{g}^k}{\|\tilde{g}^k\|} \right) < -\varepsilon_d.$$

Then there exists  $\bar{t} > 0$  such that

$$f(x^k + td^k) < \tilde{L}_\eta(t) = f(x^k) + \eta t (d^k)^\top \tilde{g}^k \quad \text{for all } t \in (0, \bar{t}).$$

In particular, if  $\mu^k$  is sufficiently small, then there is an open interval of step lengths satisfying Condition (10.1).

**PROOF.** By Proposition 10.4, we have that  $(d^k)^\top \nabla f(x^k) < 0$ . Recall that

$$\lim_{\tau \rightarrow 0} \frac{f(x^k + \tau d^k) - f(x^k)}{\tau} = (d^k)^\top \nabla f(x^k) < 0.$$

So, for any  $\hat{\eta} \in (0, 1)$ , there exists  $\bar{t} > 0$  such that

$$\frac{f(x^k + td^k) - f(x^k)}{t} < \hat{\eta} (d^k)^\top \nabla f(x^k)$$

whenever  $t \in (0, \bar{t})$ . (Later in the proof, we shall use the specific value  $\hat{\eta} = \frac{1+\eta}{2}$ , but for now we will maintain the shorter notation.) Rearranging this we have, whenever  $t \in (0, \bar{t})$ ,

$$\begin{aligned} f(x^k + td^k) - f(x^k) &< t\hat{\eta}(d^k)^\top \nabla f(x^k) \\ &= t\hat{\eta}((d^k)^\top \tilde{g}^k + (d^k)^\top (\nabla f(x^k) - \tilde{g}^k)) \\ &\leq t\hat{\eta}((d^k)^\top \tilde{g}^k + \|d^k\| \|\nabla f(x^k) - \tilde{g}^k\|) \\ &\leq t\hat{\eta}((d^k)^\top \tilde{g}^k + \|d^k\| \kappa_g(x^k) \Delta^k). \end{aligned}$$

The last inequality is simply the definition of controllable accuracy. Using the bounds  $\Delta^k < \mu^k \|\tilde{g}^k\|$  and  $\mu^k < \frac{\varepsilon_d}{\kappa_g(x^k)} \left( \frac{1-\eta}{1+\eta} \right)$ , we find

$$\begin{aligned} f(x^k + td^k) - f(x^k) &\leq t\hat{\eta}((d^k)^\top \tilde{g}^k + \|d^k\| \kappa_g(x^k) \mu^k \|\tilde{g}^k\|) \\ &\leq t\hat{\eta}\left((d^k)^\top \tilde{g}^k + \varepsilon_d \left(\frac{1-\eta}{1+\eta}\right) \|d^k\| \|\tilde{g}^k\|\right) \\ &< t\hat{\eta}\left((d^k)^\top \tilde{g}^k - \left(\frac{1-\eta}{1+\eta}\right) d^k \tilde{g}^k\right) \quad \text{by (10.4)} \end{aligned}$$

$$\begin{aligned}
&= t\hat{\eta} (d^k)^\top \tilde{g}^k \left( 1 - \left( \frac{1-\eta}{1+\eta} \right) \right) \\
&= t \left( \frac{2\eta\hat{\eta}}{1+\eta} \right) (d^k)^\top \tilde{g}^k.
\end{aligned}$$

This inequality is valid for any value of  $\hat{\eta} \in (0, 1)$ . In particular, setting  $\hat{\eta} = \frac{1+\eta}{2} \in (0, 1)$  implies that there exists  $\bar{t} > 0$  such that

$$f(x^k + td^k) - f(x^k) = t\eta(d^k)^\top \tilde{g}^k$$

whenever  $t \in (0, \bar{t})$ .

The final statement takes into account that  $\Delta^k < \mu^k \|\tilde{g}^k\|$  is established in Step 2 of the MBD algorithm.  $\square$

### 10.5. Convergence

We now provide a final convergence result for MBD. We shall require three basic assumptions.

First, and in order to use the notion of controllable accuracy of the gradient, we assume  $f \in \mathcal{C}^1$ . As model-based methods work by approximating gradients (and possibly Hessians), without this assumption the method makes very little sense.

Second, the theorem assumes the MBD algorithm has access to controllably accurate gradient approximations  $\tilde{g}_{\Delta^k}$ . Theorem 10.6 will assume that the controllable accuracy parameter  $\kappa_g$  for the models of  $f$  is actually a constant, independent of  $x^k$ . This can be relaxed, but making this assumption considerably simplifies the proof. In Chapter 9, we saw that controllable accuracy can be reasonably achieved in many situations.

Finally, the theorem assumes the existence of a scalar  $\bar{t} > 0$  such that  $t^k \|d^k\| \geq \bar{t}$  for all  $k$ . In practice, it is reasonable (and often helpful) to force the normalisation  $\|d^k\| = 1$  for all  $k$ , so this assumption can be viewed as the step-length is bounded away from 0. This is a very common assumption in practice, and often justified with arguments like “machine precision places a minimum value on  $t^k$ ” or “this is theoretical only, and in practice stopping conditions should occur long before the bound is reached”.

**THEOREM 10.6 (Convergence of the MBD Algorithm).**

Suppose  $f \in \mathcal{C}^1$  is bounded below and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$  with constant  $\kappa_g$ . Suppose that there exists  $\bar{t} > 0$  such that  $t^k \|d^k\| \geq \bar{t}$  for all  $k$ . If MBD is run with  $\epsilon_{\text{stop}} = 0$ , then

$$\lim_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0.$$

**PROOF.** The proof considers two disjoint cases.

**Case I:** (finite line-search successes) Suppose  $x^k$  is unchanged after iteration  $\bar{k}$ . Then, we must have  $\nabla f(x^{\bar{k}}) = 0$ . Indeed, if  $\nabla f(x^{\bar{k}}) \neq 0$ , then Proposition 10.3 implies for any given  $\mu^k$  eventually a model will be deemed accurate, so infinite line-search failures would imply  $\mu^k \rightarrow 0$ . Moreover, if  $\nabla f(x^{\bar{k}}) \neq 0$  and  $\mu^k \rightarrow 0$ , then Theorem 10.5 implies that once  $\mu^k$  is sufficiently small, then the line-search will succeed.

**Case II:** (infinite line-search successes) Suppose there are an infinite number of decrease steps. As  $x^k$  is only changed on line-search successes, dropping to a subsequence as necessary, without loss of generality we may assume

$$f(x^{k+1}) \leq f(x^k) + \eta t^k (d^k)^\top \tilde{g}^k \quad \text{for all } k.$$

So,  $f(x^k)$  is a nonincreasing sequence, and bounded below by assumption. Therefore,  $f(x^k)$  converges, say to  $\bar{f}$ . (Note,  $f(x^k)$  is the function value, its convergence does not necessarily imply  $x^k$  converges.) We must also have  $f(x^{k+1}) \rightarrow \bar{f}$ .

Rearranging the Armijo condition, we have

$$0 < -\eta t^k (d^k)^\top \tilde{g}^k \leq f(x^k) - f(x^{k+1}).$$

By the squeeze theorem (page 20),

$$0 \leq \lim_{k \rightarrow \infty} -\eta t^k (d^k)^\top \tilde{g}^k \leq \lim_{k \rightarrow \infty} f(x^k) - f(x^{k+1}) = \bar{f} - \bar{f} = 0.$$

So,  $\lim_{k \rightarrow \infty} t^k (d^k)^\top \tilde{g}^k = 0$ . By Step 3 of the algorithm, and the equivalent formulation (10.4), we have  $-(d^k)^\top \tilde{g}^k > \varepsilon_d \|d^k\| \|\tilde{g}^k\|$ . Again applying the squeeze theorem, leads to

$$0 \leq \lim_{k \rightarrow \infty} \varepsilon_d t^k \|d^k\| \|\tilde{g}^k\| \leq \lim_{k \rightarrow \infty} -t^k (d^k)^\top \tilde{g}^k = 0.$$

As  $t^k \|d^k\| \geq \bar{t}$  for all  $k$ , it follows that

$$\lim_{k \rightarrow \infty} \|\tilde{g}^k\| = 0.$$

Finally, using the triangle inequality, controllable accuracy, and model accuracy from Step 2, we have

$$\begin{aligned} 0 &\leq \|\nabla f(x^k)\| \leq \|\nabla f(x^k) - \tilde{g}^k\| + \|\tilde{g}^k\| \leq \kappa_g \Delta^k + \|\tilde{g}^k\| \\ &\leq \kappa_g \mu^k \|\tilde{g}^k\| + \|\tilde{g}^k\|. \end{aligned}$$

One final application of the squeeze theorem and the proof is complete,

$$0 \leq \lim_{k \rightarrow \infty} \|\nabla f(x^k)\| \leq \lim_{k \rightarrow \infty} \kappa_g \mu^k \|\tilde{g}^k\| + \|\tilde{g}^k\| = 0.$$

□

Notice that the MBD algorithm provides a different flavour of convergence result than the methods in Part 3. Specifically, the methods in Part 3 analyzed the properties of the *refined points* of the algorithm, showing that refined points exists and are critical points (assuming the objective function is Lipschitz). Conversely, the MBD algorithm shows that the gradient

converges to 0, without the need to drop to specific subsequences. An immediate corollary is that any accumulation point must be a critical point (see Exercise 10.13).

## 10.6. Additional Experiments with the Rheology Problem

We end this chapter by applying the MBD algorithm to a variant of the rheology parameter fitting optimization problem from Section 1.3.3. Since we are working with model-based methods, we focus on the smooth rheology parameter fitting problem.

**EXAMPLE 10.1.** Consider the smooth rheology parameter fitting problem from Section 1.3.3, as it was reformulated in Example 3.1:

$$\min_{x \in \mathbb{R}^3} \{\check{f}(x) : x \in [\ell, u]\}.$$

We use the 7 starting points from Example 3.5, noting that the initial function values are as follows:

$$\begin{aligned} x_{\text{GS}}^0 &= [15, & 20, & 10]^T & \hat{f}(x_{\text{GS}}^0) &\approx 21064.9, \\ x_{\text{LHS}_1}^0 &= [8.172517606, & 5.058263716, & 5.444856567]^T & \hat{g}(x_{\text{LHS}_1}^0) &\approx 42664.5, \\ x_{\text{LHS}_2}^0 &= [13.04832254, & 15.84400309, & 9.950620587]^T & \hat{g}(x_{\text{LHS}_2}^0) &\approx 15414.2, \\ x_{\text{LHS}_3}^0 &= [12.31453665, & 13.75028434, & 9.557207957]^T & \hat{g}(x_{\text{LHS}_3}^0) &\approx 12277.2, \\ x_{\text{LHS}_4}^0 &= [11.36633281, & 12.12935162, & 8.906909739]^T & \hat{g}(x_{\text{LHS}_4}^0) &\approx 17275.8, \\ x_{\text{LHS}_5}^0 &= [9.690281657, & 6.799833301, & 5.904578444]^T & \hat{g}(x_{\text{LHS}_5}^0) &\approx 83401.4, \\ x_{\text{LHS}_6}^0 &= [12.20082785, & 12.61627174, & 8.890182552]^T & \hat{g}(x_{\text{LHS}_6}^0) &\approx 18438.3. \end{aligned}$$

We apply the MBD algorithm, using  $\mu^0 = 1$ ,  $\eta = 0.05$ , and two different values for  $\Delta^0$ :  $\Delta^0 = 1$  and  $\Delta^0 = 0.001$ . For comparison, we also apply a simple implementation of a gradient descent method. To do this, we assume that the exact gradient  $g$  of  $\check{f}$  is provided (for free) every time the function is evaluated. The gradient descent algorithm can then be viewed as the MBD algorithm except with the simplex gradient replaced with the exact gradient. We ran these three algorithms from each of the seven starting points, with an additional budget of 125 and 375 function evaluations. Table 10.1 presents the results. For comparison, column “CS best” in Table 10.1 presents the best result found using the CS algorithm as detailed in Example 3.5.

As expected, 375 function evaluations outperforms 125 additional function evaluations in every case. Notice that while using  $\Delta^0 = 0.001$  generally outperforms using  $\Delta^0 = 1$ , the trend is not universal. Often the two initial values for  $\Delta^0$  behave comparably.

Comparing the MBD algorithm to the CS algorithm, we see the value of the MBD algorithm for smooth problems. Indeed, when the budget is 375  $\check{f}$ -calls, the MBD algorithm outperforms the CS algorithm in every case (often quite significantly). At a budget of 125  $\check{f}$ -calls, the MBD algorithm outperforms the CS algorithm in most cases, but loses when starting from  $x_{\text{LHS}_1}^0$  for  $\Delta^0 = 1$  and when starting from  $x_{\text{LHS}_5}^0$  for both choices of  $\Delta^0$ .

TABLE 10.1. Best function value  $\check{f}$  on the smooth rheology problem, generated using different initial points, the MBD algorithm, and a gradient descent algorithm

| Initial Point        | CS best                   |                           | MBD, $\Delta^0 = 1$       |                           | MBD, $\Delta^0 = 0.001$   |                           | Gradient Descent |                  |
|----------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|------------------|------------------|
|                      | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | 125<br>f/g-calls | 375<br>f/g-calls |
| $x_{\text{GS}}^0$    | 8033.51                   | 7878.70                   | 8236.6                    | 7836.3                    | 7849.3                    | 7249.6                    | 7649.3           | 7191.4           |
| $x_{\text{LHS}_1}^0$ | 5139.76                   | 1094.98                   | 17323.3                   | 182.3                     | 845.8                     | 192.8                     | 282.1            | 171.8            |
| $x_{\text{LHS}_2}^0$ | 5819.71                   | 5310.60                   | 5830.4                    | 5252.7                    | 5557.1                    | 4436.8                    | 5468.8           | 4198.9           |
| $x_{\text{LHS}_3}^0$ | 4657.93                   | 4325.62                   | 4439.7                    | 3428.8                    | 3931.3                    | 2846.3                    | 3220.6           | 1536.1           |
| $x_{\text{LHS}_4}^0$ | 3752.66                   | 3218.25                   | 3016.4                    | 171.8                     | 2703.2                    | 1686.8                    | 240.4            | 171.8            |
| $x_{\text{LHS}_5}^0$ | 247.21                    | 217.62                    | 5050.3                    | 184.5                     | 1626.7                    | 192.3                     | 188.7            | 171.8            |
| $x_{\text{LHS}_6}^0$ | 3691.67                   | 3382.16                   | 3417.3                    | 2445.4                    | 313.6                     | 192.9                     | 2754.7           | 395.4            |

Notice how much more effective the gradient-based method is than the MBD algorithm. In fact, in most cases using gradient descent with 125 function/gradient evaluations performs comparable to, or even better than, the MBD algorithm using 375 function evaluations. Considering that the objective function is in  $\mathbb{R}^3$  this makes sense. A single gradient approximation in  $\mathbb{R}^3$  requires 4 function evaluations, so given exact gradient information for free, we would expect an approximate speed-up of 4 to 1.

This emphasises the comment on page 6: if accurate gradients are readily available, then DFO methods should not be used. This is particular poignant, as gradient descent is generally considered a very poor gradient-based method. We shall revisit this problem further in Example 11.3.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 10.1.** Consider the function  $f(x) = e^x$ . For any  $\Delta > 0$  define the model function  $\tilde{f}_\Delta(x) := \frac{1}{2}(x + \Delta + 1)^2 - 41$ .

- a) Prove that  $\tilde{g}_\Delta(x) = \nabla \tilde{f}_\Delta(x)$  are controllably accurate approximations of  $\nabla f(x)$  at  $x = 0$ .
- b) Prove that  $\{\tilde{f}_\Delta\}_{\Delta \in (0, \bar{\Delta}]}$  is not a class of fully linear models for  $f$  at 0 for any  $\bar{\Delta}$ .
- c) Why does this not contradict Proposition 10.1?
- d) How could  $\tilde{f}_\Delta$  be adjusted so that  $\{\tilde{f}_\Delta\}_{\Delta \in (0, 1]}$  forms a class of fully linear models for  $f$  at 0.

**EXERCISE 10.2.** + Prove Proposition 10.1.

**EXERCISE 10.3.** Let  $f \in \mathcal{C}^1$ . Suppose  $\nabla f(\bar{x}) \neq 0$ .

- a) Prove that  $d = -\nabla f(\bar{x})$  is a descent direction.
- b) Let  $H$  be a symmetric positive definite matrix. Prove that  $d = -H^{-1}\nabla f(\bar{x})$  is a descent direction.
- c) Suppose in the MBD algorithm  $g^k \neq 0$ . Show that  $d^k = -g^k$  will satisfy the descent direction selection test in Step 3, regardless of  $\epsilon_d$ .
- d) Suppose in the MBD algorithm  $g^k \neq 0$ . Fix  $\epsilon_d > 0$ . Show that  $d^k = -g^k$  does not necessarily satisfy  $(d^k)^\top g^k < -\epsilon_d$ .
- e)  $H$  be a symmetric positive definite matrix. Suppose in the MBD algorithm  $g^k \neq 0$ . Show that  $d = -H^{-1}\nabla g^k$  will satisfy the descent direction selection test in Step 3, but only if  $\epsilon_d$  is sufficiently small.

**EXERCISE 10.4.** Consider the function  $f : \mathbb{R}^2 \mapsto \mathbb{R}$  defined as  $f(x) = x_1^2 + 2x_2^2$ , and for any  $\Delta > 0$ , we define the gradient approximation at  $x \in \mathbb{R}^2$

$$\tilde{g}_\Delta(x) = \frac{1}{\Delta} \begin{bmatrix} f(x + \Delta e_1) - f(x) \\ f(x + \Delta e_2) - f(x) \end{bmatrix}$$

where  $e_i$  is the  $i$ -th coordinate direction.

- a) Give a value of  $\kappa_g$  for which  $\tilde{g}_\Delta$  are controllably accurate gradient approximations with constant  $\kappa_g$ .
- b) Is there a direction  $d \in \mathbb{R}^2$  that depends on both  $x$  and  $\Delta$ , such that  $d^\top \tilde{g}_\Delta(x) < 0 < d^\top \nabla f(x)$ ?

**EXERCISE 10.5.** Create an example of showing why the Armijo parameter  $\eta$  must be strictly less than 1. Specifically, give  $f \in \mathcal{C}^2$ ,  $x^k \in \mathbb{R}^n$ , and a descent direction  $d^k$ , such that

$$f(x^k + t^k d^k) < f(x^k) + \eta t^k (d^k)^\top \nabla f(x^k)$$

has no solution for  $\eta = 1$ .

**EXERCISE 10.6.** Given an algorithm with controllably accurate gradient approximations with constant  $\kappa_g > 0$ . Let

$$\bar{\mu} = \min \left\{ \frac{\epsilon_d}{2\kappa_g}, \frac{\epsilon_d(1-\eta)}{\kappa_g(1+\eta)} \right\},$$

where  $\epsilon_d > 0$ . What values of  $\eta \in (0, 1)$  result in  $\bar{\mu} = \frac{\epsilon_d}{2\kappa_g}$ ?

**EXERCISE 10.7.** Let  $f \in \mathcal{C}^1$ . Let  $\nabla f(\bar{x}) \neq 0$ , and  $0 < \eta < 1$  be given. Suppose  $\tilde{g}_\Delta(\bar{x})$  are approximate gradients of  $f$  at  $\bar{x}$  satisfying

$$\|\nabla f(\bar{x}) - \tilde{g}_\Delta(\bar{x})\| \leq \kappa_g(\bar{x})\Delta.$$

for some  $\kappa_g(\bar{x}) > 0$ . Prove that if  $\eta$  and  $\Delta$  are sufficiently small, then there exist  $\bar{t} > 0$  such that

$$f(\bar{x} - \tilde{g}_\Delta(\bar{x})) \leq f(\bar{x}) - \eta t \|\tilde{g}_\Delta(\bar{x})\|^2$$

for all  $0 < t < \bar{t}$ .

**EXERCISE 10.8.**  $\square$  One basic method of performing a line search is the *forward-backward-tracking line search*.

**ALGORITHM 10.2.** Forward-backward-tracking line search

Given  $f \in \mathcal{C}^1$ , starting point  $x^k$ , descent direction  $d^k$ , approximate gradient  $\tilde{g}^k$ , and Armijo parameter  $\eta$

0. Initialise:

$$\begin{array}{ll} \tau = 1 & \text{initial step-length} \\ N_{\max} \in \mathbb{N} & \text{bound for evaluations allowed in line search} \end{array}$$

1. Forward-backward search:

|   |
|---|
| $\left  \begin{array}{l} \text{if } f(x^k + \tau d^k) < f(x^k) + \eta \tau (d^k)^\top \tilde{g}^k \\ \quad \text{set line-search = success} \\ \text{while } \left\{ \begin{array}{l} \tau \in \{2^0, 2^1, \dots, 2^{N_{\max}}\} \\ \text{and } f(x^k + \tau d^k) < f(x^k) + \eta \tau (d^k)^\top \tilde{g}^k \end{array} \right. \\ \quad \tau \leftarrow 2\tau \\ \text{end while} \\ \quad \tau \leftarrow \tau/2 \\ \text{otherwise } (f(x^k + \tau d^k) \geq f(x^k) + \eta \tau (d^k)^\top \tilde{g}^k) \\ \quad \text{set line-search = failure} \\ \text{while } \left\{ \begin{array}{l} \tau \in \{2^0, 2^{-1}, \dots, 2^{-N_{\max}}\} \\ \text{and } f(x^k + \tau d^k) \geq f(x^k) + \eta \tau (d^k)^\top \tilde{g}^k \end{array} \right. \\ \quad \tau \leftarrow \tau/2 \\ \text{end while} \\ \quad \text{if } f(x^k + \tau d^k) < f(x^k) + \eta \tau (d^k)^\top \tilde{g}^k, \\ \quad \quad \text{set line-search = success} \end{array} \right.$ |
|---|

2. Output:

|  |
|--|
| $\left  \begin{array}{l} \text{if line-search = success, then return } \tau \\ \text{otherwise, return "line-search failure"} \end{array} \right.$ |
|--|

Implement the forward-backward-tracking line search and test it on  $f(x) = \|x\|^4$ ,  $x^k = [1, -2, 1]^\top$ ,  $\tilde{g}^k = \nabla f(x^k)$ ,  $d^k = -\tilde{g}^k$ , and the three Armijo parameters  $\eta \in \{0.1, 0.5, 0.9\}$ .

**EXERCISE 10.9.**  $\square$  Read Exercise 10.8. Implement the forward-backward-tracking line search and test it on  $f(x) = e^{x_1} + (x_2)^2 + (x_1 x_2)^2$ ,  $x^k = [0, 2]^\top$ ,  $\tilde{g}^k = [0.5, 4.5]^\top$ ,  $d^k = -\tilde{g}^k$ , and the three Armijo parameters  $\eta \in \{0.1, 0.5, 0.9\}$ .

**EXERCISE 10.10.** Let  $f(x) = x^2$  ( $f : \mathbb{R} \mapsto \mathbb{R}$ ),  $\eta = \frac{1}{2}$  and  $\bar{x} = 1$ .

- a) Let  $d = -f'(\bar{x})$ . Find the largest value  $\bar{t}$  such that the Armijo condition,

$$f(\bar{x} + td) \leq f(\bar{x}) + \eta t d f'(\bar{x}),$$

holds for all  $0 < t < \bar{t}$ .

- b) Suppose  $\tilde{g}_\Delta(x) = 2x + \Delta$ ,  $\bar{\Delta} = 1$  and let  $d = -\tilde{g}_\Delta(\bar{x})$ . Find the largest value  $\bar{t}$  such that the Armijo condition,

$$f(\bar{x} + td) \leq f(\bar{x}) + \eta t d \tilde{g}_\Delta(\bar{x}),$$

holds for all  $0 < t < \bar{t}$ .

- c) Suppose  $\tilde{g}_\Delta(x) = 2x - \Delta$ ,  $\bar{\Delta} = 1$  and let  $d = -\tilde{g}_\Delta(\bar{x})$ .  
 Find the largest value  $\bar{t}$  such that the Armijo condition,

$$f(\bar{x} + td) \leq f(\bar{x}) + \eta t d \tilde{g}_\Delta(\bar{x}),$$

holds for all  $0 < t < \bar{t}$ .

**EXERCISE 10.11.**  $\square$  Let  $f(x) = (x_1)^2 + (x_2)^2$  ( $f : \mathbb{R}^2 \mapsto \mathbb{R}$ ),  $\eta = \frac{1}{2}$  and  $\bar{x} = [1, 1]^\top$ . Suppose  $\tilde{g}_\Delta(x) = [2x_1 + \Delta, 2x_2 - \Delta]^\top$  and let  $d = -\tilde{g}_\Delta(\bar{x})$ .

- a) Find the largest value  $\bar{t}$  such that the Armijo condition,

$$f(\bar{x} + td) \leq f(\bar{x}) + \eta t d^\top \tilde{g}_\Delta(\bar{x}),$$

holds for all  $0 < t < \bar{t}$ .

- b) Set  $\Delta = 0.9$  and plot  $L(t)$ ,  $L_\eta(t)$ ,  $\tilde{L}_\eta(t)$  and  $f(\bar{x} + td)$  for  $t$  varying from 0 to  $2\bar{t}$ .  
 c) Set  $\Delta = 0.1$  and plot  $L(t)$ ,  $L_\eta(t)$ ,  $\tilde{L}_\eta(t)$  and  $f(\bar{x} + td)$  for  $t$  varying from 0 to  $2\bar{t}$ .

**EXERCISE 10.12.**  $\square$  Let  $f(x) = e^{x_1-1} + 2e^{x_2} + x_1 x_2$  ( $f : \mathbb{R}^2 \mapsto \mathbb{R}$ ),  $\eta = \frac{1}{2}$  and  $\bar{x} = [1, 0]^\top$ . Suppose  $\tilde{g}(\bar{x}) = [-0.2, 2.1]^\top$  and let  $d = -\tilde{g}(\bar{x})$ .

- a) Numerically approximate the largest value  $\bar{t}$  such that the Armijo condition,

$$f(\bar{x} + td) \leq f(\bar{x}) + \eta t d^\top \tilde{g}(\bar{x}),$$

holds for all  $0 < t < \bar{t}$ .

- b) Plot  $L(t)$ ,  $L_\eta(t)$ ,  $\tilde{L}_\eta(t)$  and  $f(\bar{x} + td)$  for  $t$  varying from 0 to  $2\bar{t}$ .

**EXERCISE 10.13.** Suppose  $f \in \mathcal{C}^1$  is bounded below and  $\tilde{g}_\Delta$  are controllably accurate gradient approximations of  $\nabla f$  with constant  $\kappa_g$ . Suppose that there exists  $\bar{t} > 0$  such that  $t^k \|d^k\| \geq \bar{t}$  for all  $k$ . Prove that if  $\hat{x}$  is an accumulation point of the MBD algorithm, then  $\nabla f(\hat{x}) = 0$ .

---

□

**Chapter 10 Project: Apply MBD to the Smooth Rheology Problem.** Implement the MBD algorithm using the simplex gradient and one other method creating controllably accurate gradient approximations. Apply the MBD algorithm on the smooth rheology problem (from Section 1.3.3). Explain how (and why) your results differ from those in Table 10.1.

# *Model-Based Trust Region*

While the line-search-based method of Chapter 10 is quick and easy to understand, it does not exploit the full power of a model function. In essence, the model-based descent algorithm for unconstrained optimization only uses gradient approximations to confirm descent directions. This can be seen in the convergence analysis of Chapter 10, where only *controllably accurate gradient approximations* are used. In particular, convergence does not even require that the function values  $f(x^k)$  be estimated by a model.

In the present chapter, we will return to the notion of *fully linear models*, and make use of the quality of the function value approximations that fully linear models provide. The *model-based trust region* (MBTR) algorithm solves the trust region subproblem of minimising the model  $\tilde{f}_{\Delta^k}$  of the objective function at iteration  $k$  within a closed ball of radius  $\Delta^k$ . This ball is called a *trust region*, i.e., a region in which the model is felt to be an accurate approximation of the true objective function  $f$ . The solution of this subproblem will be denoted by  $x_{\text{tmp}}^k$ , and referred to as a *candidate point*. The candidate point is checked for quality and either, the new point is adopted, the model is updated, the trust region is updated, or some combination of these outcomes is applied.

As in Chapter 10, we focus here on the optimization algorithm, having already covered methods for building models in Chapter 9.

### 11.1. The MBTR Algorithm

Model-based methods can be broadly split into two groups. Descent-based methods (covered in Chapter 10) use the model gradient (and possibly Hessian) information to determine a descent direction of the true function and then perform a line-search in that direction. Trust region methods take a more direct approach by minimising the model function over a trust region. The model can be updated and the trust region can be expanded or shrunk based on the results of the previous iteration.

A pseudo-code for the model-based trust region (MBTR) algorithm is next.

---

#### ALGORITHM 11.1. Model-based trust region (MBTR)

---

Given  $f \in \mathcal{C}^1$ , starting point  $x^0$  and fully linear models  $\tilde{f}_\Delta$  of  $f$

0. Initialise:

|  |   |
|--|---|
| $\Delta^0 \in (0, \infty)$               | initial trust region radius   |
| $\tilde{f}^0$                            | initial model (a fully linear model of $f$ on $B_{\Delta^0}(x^0)$ ) |
| $\mu$                                    | model accuracy parameter  |
| $\eta \in (0, 1)$                        | sufficient decrease test parameter                                  |
| $\gamma \in (0, 1)$                      | trust region update parameter                                       |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance  |
| $k \leftarrow 0$                         | iteration counter   |

1. Model building and accuracy check:

|   |
|---|
| if $\Delta^k > \mu \ \nabla \tilde{f}^k(x^k)\ $ or $\tilde{f}^k$ is not a |
| fully linear model of $f$ on $B_{\Delta^k}(x^k)$                          |
| decrease $\Delta^{k+1} = \gamma \Delta^k$ , set $x^{k+1} = x^k$           |
| use $\tilde{f}_\Delta$ to create a fully                                  |
| linear model $\tilde{f}^{k+1}$ of $f$ on $B_{\Delta^{k+1}}(x^k)$          |
| increment $k \leftarrow k + 1$ and go to 1                                |

2. Stopping criterion:

|   |
|---|
| if $\Delta^k < \epsilon_{\text{stop}}$ and $\ \nabla \tilde{f}^k(x^k)\  < \epsilon_{\text{stop}}$ |
| declare algorithm success; stop   |

3. Trust region subproblem:

|  |
|--|
| solve (or approximate) $x_{\text{tmp}}^k \in \underset{x}{\operatorname{argmin}} \{\tilde{f}^k(x) : x \in B_{\Delta^k}(x^k)\}$ |
|--|

4. Candidate test and trust region update:

evaluate  $f(x_{\text{tmp}}^k)$  and compute the ratio

$$\rho^k = \frac{f(x^k) - f(x_{\text{tmp}}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)}$$

if  $\rho^k > \eta$  (sufficient decrease exists), declare iterate success and

let  $x^{k+1}$  be any point such that  $f(x^{k+1}) \leq f(x_{\text{tmp}}^k)$

increase next trust region  $\Delta^{k+1} = \gamma^{-1} \Delta^k$

create  $\tilde{f}^{k+1}$  using  $\tilde{f}^k$  and  $x^{k+1}$

otherwise  $\rho^k \leq \eta$ , declare iterate failure and

set  $x^{k+1} = x^k$ , decrease trust region  $\Delta^{k+1} = \gamma \Delta^k$

and keep

$$\tilde{f}^{k+1} = \tilde{f}^k$$

increment  $k \leftarrow k + 1$  and go to 1

Our primary goal in this chapter is to prove convergence of MBTR by showing that it produces a sequence of trial points  $x^k$  that satisfies  $\|\nabla f(x^k)\| \rightarrow 0$ . Before moving towards this goal, we take a few moments to examine some of the subtleties of the MBTR algorithm.

We begin by commenting on the notation used within the MBTR algorithm (and the remainder of this chapter). Notice that the input of algorithm MBTR includes fully linear models  $\tilde{f}_\Delta$  of  $f$ . This means that for any value of  $\Delta$ , and for any  $x$ , the algorithm can have access to approximations of  $f$  and of  $\nabla f$  in a ball or radius  $\Delta$  around any  $x$ , as described in Section 9.1. This is used in Step 1 of the algorithm to construct the models. Tentative models are constructed at the end of Step 4, but are revised in Step 1 if they do not satisfy the model accuracy checks.

Once that the algorithm moves from Step 1 to Step 2, the resulting model is denoted by  $\tilde{f}^k$ . This notation is less precise, but much lighter to manipulate than  $\tilde{f}_{\Delta^k}^k$ . To summarise, when we will talk about fully linear models we will write  $\tilde{f}_\Delta$ , and when we will refer to the model at iteration  $k$ , we will write  $\tilde{f}^k$ .

Like MBD, the MBTR algorithm performs a model accuracy check (Step 1 in MBTR, Step 2 in MBD). While the language is slightly different, the accuracy test is the same, and its purpose is the same. Also like MBD, the MBTR algorithm terminates successfully based on the radius of the trust region  $\Delta^k$  and the gradient of the model function  $\nabla \tilde{f}^k(x^k)$ , in fact, the stopping tests are identical in both algorithms. These similarities will make analysis of Steps 1 and 2 fairly easy for the MBTR algorithm.

The major difference between MBD and MBTR appears in Step 3, where instead of performing a line-search, the MBTR algorithm seeks an approximate minimiser of the trust region subproblem,

$$(11.1) \quad x_{\text{tmp}}^k \in \operatorname{argmin}_x \{\tilde{f}^k(x) : x \in B_{\Delta^k}(x^k)\}.$$

It is not necessary to solve this problem exactly, but it is necessary to solve this problem reasonably well. How well this problem must be solved will be discussed in Section 11.3.

Figure 11.1 illustrates how the MBTR algorithm finds the candidate points. The main part of the figure shows contour plot of a two-variable objective function  $f$ . On the left plot, the disk represents the trust region  $B_{\Delta^k}(x^k)$  and shows contour plot of the fully linear model  $\tilde{f}^k$  in the vicinity of the current iterate  $x^k$ . The candidate point  $x_{\text{tmp}}^k$  is obtained by approximately minimising the model function over the trust region. A new model is built around  $x^{k+1} = x_{\text{tmp}}^k$ , as illustrated in the right plot.

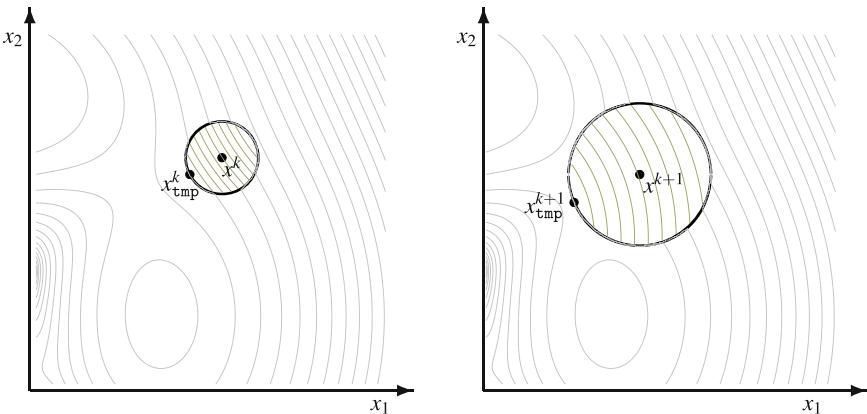


FIGURE 11.1. Illustration of a MBTR subproblems in  $\mathbb{R}^2$

In MBTR, the quality of the temporary candidate point  $x_{\text{tmp}}^k$  is assessed using the accuracy ratio of the actual reduction over the reduction predicted by the model:

$$(11.2) \quad \rho^k := \frac{f(x^k) - f(x_{\text{tmp}}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)}.$$

If this ratio is large (i.e.,  $\rho^k$  is greater than some positive constant  $\eta$ ), then the true decrease in objective function value is large when compared against the predicted decrease in function value. This is good, so the candidate point is selected as the next iterate. If  $\rho^k$  is small (i.e.,  $\rho^k \leq \eta$ ), then the true function values did not decrease as much as the model predicted. This suggests that the model needs improving, so the candidate point is not accepted and the trust region radius is decreased.

Returning to Figure 11.1, the accuracy ratio  $\rho^k$  associated with the left plot is close to 1, indicating that the model accurately represents the function  $f$ . Hence,  $x_{\text{tmp}}^k$  is accepted and the trust region radius is increased. On the

right plot, the accuracy ratio  $\rho^{k+1} < \eta$  is small, which suggest that the new model does not adequately represent the objective function. Step 4 will reject  $x_{\text{tmp}}^{k+1}$  and decrease the trust region radius.

Finally, like the MBD algorithm, MBTR is a local optimization method and uses a flexibly update step in order to break free of local minimisers. The comments of Section 10.3 can be applied directly to the MBTR algorithm.

## 11.2. Model Checks and Stopping Conditions

The MBTR and MBD algorithms employ similar stopping conditions and model accuracy checks. In this section, we briefly confirm that these stopping criteria and model accuracy checks behave well. We begin by showing that the MBTR algorithm cannot get stuck in an infinite loop in Step 1.

**PROPOSITION 11.1.**

Suppose  $f \in \mathcal{C}^1$ , and  $\tilde{f}_\Delta$  are fully linear models of  $f$ . Then MBTR can only replace  $\tilde{f}^k$  a finite number of times in a row, i.e., MBTR always exits the loop in Step 1.

**PROOF.** Note that fully linear models imply controllable accuracy, so the proof of Proposition 10.3 applies with the obvious changes in terminology.  $\square$

In MBD, it was suggested that if the model is inaccurate, then decreasing  $\Delta^k$  more aggressively may be advantageous. Similarly, in Step 1 of MBTR, instead of setting  $\Delta^{k+1} = \gamma\Delta^k$ , some prefer a more complex rule such as

$$\Delta^{k+1} = \min \left\{ \gamma\Delta^k, \frac{\mu}{1+\mu} \|\nabla \tilde{f}^k(x^k)\| \right\}.$$

The algorithm uses a single parameter  $\gamma$  to control both increase and decrease of the trust region size:  $\Delta^{k+1} = \gamma^{-1}\Delta^k$  for increase and  $\Delta^{k+1} = \gamma\Delta^k$  for decrease. One common improvement on this presentation is to use two distinct parameters,  $\Delta^{k+1} = \gamma_{inc}\Delta^k$  for increase and  $\Delta^{k+1} = \gamma_{dec}\Delta^k$  for decrease. This allows for greater flexibility and does not impact the proof of convergence. However, it does introduce some unnecessary notation, which we chose to avoid.

Turning to stopping conditions, we see that the analysis is again unchanged from Chapter 10.

**PROPOSITION 11.2.**

Suppose  $f \in \mathcal{C}^1$ , and  $\tilde{f}_\Delta$  are fully linear models of  $f$ . If, at iteration  $k$ , MBTR stops at Step 2, then

$$\|\nabla f(x^k)\| \leq (\kappa_g(x^k) + 1)\epsilon_{\text{stop}},$$

where  $\kappa_g(x^k)$  is the fully linear parameter for the models of  $f$  at  $x^k$ .

**PROOF.** Except for changing the words “controllable accuracy” to “fully linear”, the proof is unchanged from Proposition 10.2.  $\square$

### 11.3. Solving the Trust Region Subproblem

Step 3 of MBTR states that we must either solve *or approximate* the minimisation of the model function restricted to the trust region. In this section we explore what is meant by “*approximate*”. In particular, we introduce the notion of a *Cauchy point*.

**DEFINITION 11.1** (Cauchy Step, Point, and Value).

The Cauchy step  $t_c$  for the model functions  $\tilde{f}_\Delta \in \mathcal{C}^1$  at the point  $\hat{x} \in \mathbb{R}^n$  in the search radius  $\Delta > 0$  is the solution to

$$t_c \in \operatorname{argmin}_{t \geq 0} \{\tilde{f}_\Delta(\hat{x} - t\tilde{g}_\Delta) : \hat{x} - t\tilde{g}_\Delta \in B_\Delta(\hat{x})\}$$

where  $\tilde{g}_\Delta = \nabla \tilde{f}_\Delta(\hat{x}) \in \mathbb{R}^n$ . The corresponding Cauchy point is  $x_c = \hat{x} - t_c \tilde{g}_\Delta$ , and the Cauchy value is the function value at the Cauchy point,  $\tilde{f}_\Delta(x_c)$ .

In other words, the Cauchy point is the solution one would obtain if a perfect line-search in the direction of steepest descent from  $\hat{x}$  were used to approximate the solution to the trust region subproblem. If  $\tilde{f}_\Delta$  is reasonably well behaved, then the Cauchy point can be either computed exactly or accurately approximated using standard line-search techniques.

**EXAMPLE 11.1.** Suppose  $\Delta^k = 1/\sqrt{5}$  and  $\tilde{f}^k(x) := \tilde{f}_{\Delta^k}(x) = x_1 x_2$  is a model function at  $x^k = [-1, 2]^\top$ . To find the exact Cauchy step, point, and value, we first require to compute

$$\tilde{g}^k := \nabla \tilde{f}^k(x^k) = [x_2^k, x_1^k]^\top = [2, -1]^\top.$$

The objective in determining the Cauchy step is the minimisation of

$$\tilde{f}^k(x^k - t\tilde{g}^k) = f(-1 - 2t, 2 + t) = (-1 - 2t)(2 + t),$$

subject to the constraints the  $t \geq 0$  and  $x^k - t\tilde{g}^k \in B_{\Delta^k}(x^k)$ , which reduces to  $0 \leq t \leq 1/5$ . Combining our objective function and constraints, we seek

$$t_c \in \operatorname{argmin}_t \{(-1 - 2t)(2 + t) : 0 \leq t \leq 1/5\},$$

which is solved at  $t_c = 1/5$ . The Cauchy point is therefore  $x_c = x^k - t_c \tilde{g}^k = [-7/5, 11/5]^\top$ , and the Cauchy value is  $\tilde{f}^k(x_c) = -77/25$ .

Up to now, we have assumed the models are fully linear, but have made no assumption on the form the models functions take. In order to simplify the analysis, from this point on, we shall assume that the model functions  $\tilde{f}_\Delta$  are quadratics; i.e.,

$$(11.3) \quad \tilde{f}_\Delta(x) = \alpha_0 + \alpha^\top x + \frac{1}{2}x^\top Hx,$$

where  $\alpha_0 \in \mathbb{R}$ ,  $\alpha \in \mathbb{R}^n$ , and  $H \in \mathbb{R}^{n \times n}$  with  $H = H^\top$ . All of the methods in Chapter 9 create models of this form. With this notation the gradient and Hessian of the quadratic model at  $x$  are

$$\nabla \tilde{f}_\Delta(x) = \alpha + Hx \quad \text{and} \quad \nabla^2 \tilde{f}_\Delta(x) = H.$$

Using quadratic models, it is possible to obtain an upper bound on the Cauchy value.

**LEMMA 11.3 (Approximating the Cauchy Value).**

Consider  $\hat{x} \in \mathbb{R}^n$ ,  $\Delta > 0$ , and let  $\tilde{f}_\Delta$  be a quadratic model of the form (11.3). Let  $x_C$  be the Cauchy point for  $\tilde{f}_\Delta$  at  $\hat{x} \in \mathbb{R}^n$  using  $\Delta$ . Then

$$(11.4) \quad \tilde{f}_\Delta(x_C) \leq \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \|\tilde{g}\| \min \left\{ \frac{\|\tilde{g}\|}{\|H\|}, \Delta \right\},$$

where  $\tilde{g} = \nabla \tilde{f}_\Delta(\hat{x}) = \alpha + H\hat{x}$ , and  $\frac{\|\tilde{g}\|}{\|H\|}$  is taken to be  $\infty$  if  $H = 0$ .

**PROOF.** The result is clearly true when  $\tilde{g} = 0$  because the Cauchy value is certainly less than or equal to  $\tilde{f}(\hat{x})$ , so we now assume  $\tilde{g} \neq 0$ .

For ease of notation, consider the quadratic function  $\tilde{\psi} : \mathbb{R} \rightarrow \mathbb{R}$  that returns the model function value at a step of length  $t$  in the direction  $-\tilde{g}$  from  $\hat{x}$ :

$$\begin{aligned} \tilde{\psi}(t) &= \tilde{f}_\Delta(\hat{x} - t\tilde{g}) &= \alpha_0 + \alpha^\top(\hat{x} - t\tilde{g}) + \frac{1}{2}(\hat{x} - t\tilde{g})^\top H(\hat{x} - t\tilde{g}) \\ &= \tilde{f}_\Delta(\hat{x}) - t\tilde{g}^\top(\alpha + H\hat{x}) + \frac{1}{2}t^2\tilde{g}^\top H\tilde{g} \\ &= \tilde{f}_\Delta(\hat{x}) - t\tilde{g}^\top\tilde{g} + \frac{1}{2}t^2\tilde{g}^\top H\tilde{g}. \end{aligned}$$

The Cauchy value may be written as

$$(11.5) \quad \tilde{f}_\Delta(x_C) = \min_t \left\{ \tilde{\psi}(t) : 0 \leq t \leq \frac{\Delta}{\|\tilde{g}\|} \right\}.$$

We break the analysis into two disjoint cases: the quadratic function  $\tilde{\psi}$  is either concave or strictly convex.

**Case I ( $\tilde{\psi}$  Is Concave)** Suppose  $\tilde{g} \neq 0$  and  $\tilde{g}^\top H\tilde{g} \leq 0$ . Then the minimum of the quadratic function occurs at one of the endpoints of the interval  $[0, \Delta/\|\tilde{g}\|]$ . Since  $\tilde{\psi}(0) = \tilde{f}_\Delta(\hat{x})$  and

$$\begin{aligned} \tilde{\psi}\left(\frac{\Delta}{\|\tilde{g}\|}\right) &= \tilde{f}_\Delta(\hat{x}) - \frac{\Delta}{\|\tilde{g}\|}\tilde{g}^\top\tilde{g} + \frac{1}{2}\left(\frac{\Delta}{\|\tilde{g}\|}\right)^2\tilde{g}^\top H\tilde{g} \\ &= \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\|\Delta + \frac{1}{2}\left(\frac{\Delta}{\|\tilde{g}\|}\right)^2\tilde{g}^\top H\tilde{g} \\ &\leq \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\|\Delta \quad \text{since } \left(\frac{\Delta}{\|\tilde{g}\|}\right)^2\tilde{g}^\top H\tilde{g} \leq 0 \end{aligned}$$

then the Cauchy step is  $t_c = \frac{\Delta}{\|\tilde{g}\|}$ , with Cauchy point  $x_c = \hat{x} + \Delta \frac{d}{\|\tilde{g}\|}$  with  $f(x_c) \leq \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \Delta$ .

**Case II ( $\tilde{\psi}$  Is Strictly Convex)** Suppose  $\tilde{g}^\top H \tilde{g} > 0$ : Equation (11.5) is the minimisation of a strictly convex quadratic whose critical point is at

$$(11.6) \quad \bar{t} = \frac{\tilde{g}^\top \tilde{g}}{\tilde{g}^\top H \tilde{g}} = \frac{\|\tilde{g}\|^2}{\tilde{g}^\top H \tilde{g}} > 0.$$

Therefore if  $\bar{t} \leq \frac{\Delta}{\|\tilde{g}\|}$ , then  $\bar{t}$  is the unique minimiser to problem (11.5), and the Cauchy step is  $t_c = \bar{t}$ , with Cauchy point  $x_c = \hat{x} + t_c d$  and

$$\begin{aligned} \tilde{f}_\Delta(x_c) &= \tilde{\psi}(t_c) = \tilde{f}_\Delta(\hat{x}) - \frac{\|\tilde{g}\|^4}{\tilde{g}^\top H \tilde{g}} + \frac{1}{2} \left( \frac{\|\tilde{g}\|^2}{\tilde{g}^\top H \tilde{g}} \right)^2 \tilde{g}^\top H \tilde{g} \\ &= \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \frac{\|\tilde{g}\|^4}{\tilde{g}^\top H \tilde{g}} \\ &\leq \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \frac{\|\tilde{g}\|^2}{\|H\|} \quad \text{since } \tilde{g}^\top H \tilde{g} \leq \|\tilde{g}\|^2 \|H\|. \end{aligned}$$

The last subcase to consider is when  $\bar{t} > \frac{\Delta}{\|\tilde{g}\|}$ . The minimum of the quadratic function occurs at the Cauchy step  $t_c = \frac{\Delta}{\|\tilde{g}\|}$ , with Cauchy point  $x_c = \hat{x} + t_c d$  and

$$\begin{aligned} \tilde{f}_\Delta(x_c) &= \tilde{\psi}(t_c) = \tilde{f}_\Delta(\hat{x}) - \frac{\Delta}{\|\tilde{g}\|} \|\tilde{g}\|^2 + \frac{1}{2} \left( \frac{\Delta}{\|\tilde{g}\|} \right)^2 \tilde{g}^\top H \tilde{g} \\ &= \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \Delta + \frac{1}{2} \left( \frac{\Delta}{\|\tilde{g}\|} \right)^2 \tilde{g}^\top H \tilde{g} \\ &= \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \Delta + \frac{1}{2} \frac{\Delta^2}{t} \quad \text{by Equation (11.6)} \\ &< \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \Delta + \frac{1}{2} \Delta \|\tilde{g}\| \quad \text{because } \bar{t} > \frac{\Delta}{\|\tilde{g}\|}. \end{aligned}$$

In summary, combining Cases I and II, we know that the Cauchy value satisfies

$$\tilde{f}_\Delta(x_c) \leq \begin{cases} \tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \Delta & \text{when } \tilde{g}^\top H \tilde{g} \leq 0, \\ \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \|\tilde{g}\| \frac{\|\tilde{g}\|}{\|H\|} & \text{when } \tilde{g}^\top H \tilde{g} > 0 \text{ and } \bar{t} \leq \frac{\Delta}{\|\tilde{g}\|}, \\ \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \|\tilde{g}\| \Delta & \text{when } \tilde{g}^\top H \tilde{g} > 0 \text{ and } \bar{t} > \frac{\Delta}{\|\tilde{g}\|}. \end{cases}$$

Finally, since  $\tilde{f}_\Delta(\hat{x}) - \|\tilde{g}\| \leq \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \|\tilde{g}\|$ , we conclude

$$\tilde{f}_\Delta(x_c) \leq \tilde{f}_\Delta(\hat{x}) - \frac{1}{2} \|\tilde{g}\| \min \left\{ \frac{\|\tilde{g}\|}{\|H\|}, \Delta \right\},$$

as desired.  $\square$

**EXAMPLE 11.2.** Consider again  $\tilde{f}^k(x) = x_1 x_2$ ,  $x^k = [-1, 2]^\top$ ,  $\tilde{g}^k = [2, -1]^\top$  and  $\Delta^k = 1/\sqrt{5}$  (from Example 11.1). To apply Lemma 11.3, we note that

$\|\tilde{g}^k\| = \sqrt{5}$ , and

$$\nabla^2 \tilde{f}^k(x) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{so } \|H\| = 1 \quad (\text{using the induced } \ell_2 \text{ norm}).$$

The right-hand side of Equation (11.4) is therefore

$$\tilde{f}^k(x^k) - \frac{1}{2}\|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H\|}, \Delta^k \right\} = -2 - \frac{\sqrt{5}}{2} \min \left\{ \frac{\sqrt{5}}{1}, \frac{1}{\sqrt{5}} \right\} = -2.5.$$

Comparing to the true Cauchy value obtained in Example 11.1, we indeed have  $-77/25 = -3.08 \leq -2.5$ .

---

## 11.4. Convergence

In this section we will rely on the results from Section 11.3, and assume that at every iteration of MBTR the model functions  $\tilde{f}^k$  are quadratic;<sup>1</sup> i.e.,

$$(11.7) \quad \tilde{f}^k(x) = \alpha_0^k + (\alpha^k)^\top x + \frac{1}{2}x^\top H^k x,$$

where  $\alpha_0^k \in \mathbb{R}$ ,  $\alpha^k \in \mathbb{R}^n$ , and  $H^k \in \mathbb{R}^{n \times n}$  with  $H^k = (H^k)^\top$ . These models are trusted in the ball of radius  $\Delta^k > 0$  centred at  $x^k$ . The gradient and Hessian of the quadratic model at  $x^k$  simplify to

$$\tilde{g}^k := \nabla \tilde{f}^k(x^k) = \alpha^k + H^k x^k \quad \text{and} \quad \nabla^2 \tilde{f}^k(x^k) = H^k.$$

In addition, we assume that the norm of the matrices  $H^k$  are uniformly bounded above by a constant  $\kappa_H$ .

Lemma 11.3 ensures that the Cauchy point  $x_C^k$  at iteration  $k$  (at the point  $x^k \in \mathbb{R}^n$  in the search radius  $\Delta^k > 0$ ) satisfies

$$(11.8) \quad \tilde{f}^k(x_C^k) \leq \tilde{f}^k(x^k) - \frac{1}{2}\|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\}.$$

Therefore, to approximately solve the trust region subproblem, we fix an accuracy parameter  $\mu_C \in (0, 1)$  and aim to find an approximate minimiser  $x_{\text{tmp}}^k$  of the subproblem such that

$$(11.9) \quad \tilde{f}^k(x_{\text{tmp}}^k) \leq \tilde{f}^k(x^k) + \frac{1}{2}\mu_C \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\},$$

where  $\frac{\|\tilde{g}^k\|}{\|H^k\|}$  is taken as  $\infty$  if  $H^k = 0$ . That is, we aim to get at least a fraction of the potential improvement that would be given by the Cauchy point. In practice  $\mu_C$  can be taken quite large (say  $\mu_C = 0.95$ ), as the proof of Lemma 11.3 is quite constructive in nature, so finding a point to satisfy Equation (11.9) is theoretically achievable even using the value  $\mu_C = 1$ .

Our next step in proving convergence is to show that whenever  $\nabla f(x^k) \neq 0$ , the MBTR algorithm eventually produces a successful iterate. We have

---

<sup>1</sup>Recall, at iteration  $k$  of MBTR, the model function is constructed with the radius  $\Delta^k$  and is centred around the point  $x^k$ . To lighten the notation, we drop the subscript and simply write  $\tilde{f}^k$  rather than  $\tilde{f}_{\Delta^k}^k$ .

already established that, unless  $\nabla f(x^k) = 0$  the MBTR algorithm will eventually pass the model checks and move out of Step 1 (Proposition 11.1).

**THEOREM 11.4 (MBTR Has Successful Iterates).**

Suppose  $f \in \mathcal{C}^1$  is bounded below, and  $\tilde{f}_\Delta$  are fully linear models of  $f$  with constants  $\kappa_f$  and  $\kappa_g$  of the quadratic form (11.7) and suppose there exists  $\kappa_H > 0$  such that  $\|H^k\| \leq \kappa_H$  for all  $k$ . Let  $\mu_c \in (0, 1)$  and suppose that all test points  $x_{\text{tmp}}^k$  produced by the MBTR algorithm satisfy Equation (11.9). Suppose that  $\nabla f(x^k) \neq 0$ . If  $\Delta^k$  is sufficiently small, then  $\rho^k > \eta$ . Specifically, setting  $\tilde{g}^k := \nabla \tilde{f}^k(x^k)$ , we have

$$(11.10) \quad \Delta^k < \min \left\{ \frac{\|\tilde{g}^k\|}{\kappa_H}, \frac{\mu_c \|\tilde{g}^k\|}{4\kappa_f} (1 - \eta) \right\} \Rightarrow \rho^k > \eta,$$

In particular, if  $\nabla f(x^k) \neq 0$ , the MBTR algorithm eventually produces a successful iterate.

**PROOF.** By Equation (11.10) and  $\|H^k\| \leq \kappa_H$ , we see that  $\Delta^k < \frac{\|\tilde{g}^k\|}{\kappa_H} \leq \frac{\|\tilde{g}^k\|}{\|H^k\|}$  and therefore, Equation (11.9) becomes

$$(11.11) \quad 0 < \frac{1}{2}\mu_c \|\tilde{g}^k\| \Delta^k = \frac{1}{2}\mu_c \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\} \leq \tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k).$$

Now consider the value  $|\rho^k - 1|$ , where  $\rho^k$  is the algorithmic parameter that represents the ratio of the actual function value decrease over the predicted model decrease value:

$$\begin{aligned} |\rho^k - 1| &= \left| \frac{f(x^k) - f(x_{\text{tmp}}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)} - 1 \right| = \frac{|f(x^k) - \tilde{f}^k(x^k) + \tilde{f}^k(x_{\text{tmp}}^k) - f(x_{\text{tmp}}^k)|}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|} \\ &\leq \frac{|f(x^k) - \tilde{f}^k(x^k)|}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|} + \frac{|\tilde{f}^k(x_{\text{tmp}}^k) - f(x_{\text{tmp}}^k)|}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|}. \end{aligned}$$

Since  $\tilde{f}$  is fully linear, both numerators in the last expression are bounded above by  $\kappa_f(\Delta^k)^2$ , this yields

$$|\rho^k - 1| \leq \frac{\kappa_f(\Delta^k)^2}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|} + \frac{\kappa_f(\Delta^k)^2}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|} = \frac{2\kappa_f(\Delta^k)^2}{|\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)|}.$$

Applying estimate (11.11) ensures that

$$|\rho^k - 1| \leq \frac{2\kappa_f(\Delta^k)^2}{\frac{1}{2}\mu_c \|\tilde{g}^k\| \Delta^k} = \frac{4\kappa_f \Delta^k}{\mu_c \|\tilde{g}^k\|}.$$

Finally, applying  $\Delta^k < \frac{\mu_c \|\tilde{g}^k\|}{4\kappa_f} (1 - \eta)$  we have that  $|\rho^k - 1| < 1 - \eta$ , which implies  $1 - \rho^k < 1 - \eta$ , and therefore  $\rho^k > \eta$ .

To prove the final statement first note that  $\Delta^{k+1} \leq \tilde{\Delta}_{\text{tmp}}^{k+1} = \gamma \Delta^k$  whenever an iterate is declared a failure. In particular, an infinite sequence of failures will drive  $\Delta^{k+1}$  to 0. Whereas,  $\Delta^k \rightarrow 0$  implies (by definition of fully linear models)  $\|\tilde{g}^k\| \rightarrow \|\nabla f(x^k)\| > 0$ . Hence, eventually Equation (11.10) will be satisfied, and the next iterate will be successful.  $\square$

At this point, the remainder of the convergence proof follows similar arguments as those in Chapter 10. If the MBTR algorithm stops, then Proposition 11.2 provides a bound on the gradient at the final iterate. If  $x^k$  is unchanged after a finite number of steps, then we must have  $\nabla f(x^k) = 0$  (or we contradict Theorem 11.4). If an infinite series of successful iterates occurs, then our model accuracy and descent tests will ensure that  $\|\nabla f(x^k)\| \rightarrow 0$ . A formal proof appears in Theorem 11.5, next.

**THEOREM 11.5 (Convergence of the MBTR Algorithm).**

Suppose  $f \in \mathcal{C}^1$  is bounded below, and  $\tilde{f}_\Delta$  are fully linear models of  $f$  with constants  $\kappa_f$  and  $\kappa_g$  of the quadratic form (11.7) and suppose there exists  $\kappa_H > 0$  such that  $\|H^k\| \leq \kappa_H$  for all  $k$ . Let  $\mu_c \in (0, 1)$  and suppose that all test points  $x_{\text{tmp}}^k$  produced by MBTR satisfy Equation (11.9). If MBTR is run with  $\epsilon_{\text{stop}} = 0$ , then

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0.$$

**PROOF.** We first consider the situation where there are a finite number of trust region successes and  $x^k$  remains unchanged after iterate  $\bar{k}$ . Then, we must have  $\nabla f(x^k) = 0$  and the proof is complete. Indeed, if  $\nabla f(x^k) \neq 0$ , then Theorem 11.4 implies we will eventually have a successful iterate.

Second, we consider the situation where there are infinitely many successful iterations. As  $x^k$  is only updated on successful iterations, dropping to a subsequence as necessary, we may assume all iterations are successful.

On successful iterations we have

$$\rho^k = \frac{f(x^k) - f(x_{\text{tmp}}^k)}{\tilde{f}^k(x^k) - \tilde{f}^k(x_{\text{tmp}}^k)} > \eta.$$

Both the parameter  $\eta$  and the denominator are positive (by Equation (11.9)), so  $f(x^k) > f(x_{\text{tmp}}^k) \geq f(x^{k+1})$ . As  $f$  is bounded below, this implies that the sequence  $f(x^k)$  converges. Consequently, Equation (11.9) and the squeeze theorem give

$$\begin{aligned} 0 &= \lim_{k \rightarrow \infty} f(x^k) - f(x^{k+1}) \geq \lim_{k \rightarrow \infty} f(x^k) - f(x_{\text{tmp}}^k) \\ &\geq \lim_{k \rightarrow \infty} \eta \left( \frac{1}{2} \mu_c \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\} \right) \geq 0, \end{aligned}$$

where  $\tilde{g}^k := \nabla \tilde{f}^k(x^k)$ . This implies

$$(11.12) \quad \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\} \rightarrow 0.$$

This presents three possibilities, either  $\liminf_{k \rightarrow \infty} \|\tilde{g}^k\| = 0$ , or  $\liminf_{k \rightarrow \infty} \Delta^k = 0$ , or both. We claim it is both.

If  $\liminf_{k \rightarrow \infty} \|\tilde{g}^k\| = 0$ , then  $\Delta^k < \|\tilde{g}^k\|/\kappa_H$  (by Equation (11.10)) implies that  $\liminf_{k \rightarrow \infty} \|\Delta^k\| = 0$ .

By way of contradiction, suppose  $\liminf_{k \rightarrow \infty} \Delta^k = 0$ , but there exists a scalar  $\beta > 0$  such that  $\|\tilde{g}^k\| \geq \beta$  for every  $k$ . There are two places where  $\Delta^k$  can be decreased: Step 1 and Step 4. In both cases we shall establish a positive lower bound on  $\Delta^k$  based on  $\beta$ , thereby contradicting  $\liminf_{k \rightarrow \infty} \Delta^k = 0$ .

We begin with Step 1. Following the same arguments as Proposition 10.3, if

$$\Delta^k \leq \frac{\mu}{(1 + \mu\kappa_g)} \|\nabla f(x^k)\|,$$

then Step 1 will be successful. Since  $\Delta^k$  is decreased by a factor of  $\gamma$  when unsuccessful, the smallest  $\Delta^k$  can become is  $\gamma \frac{\mu}{(1 + \mu\kappa_g)} \|\nabla f(x^k)\|$ . That is, when Step 1 is passed, we must have

$$(11.13) \quad \Delta^k \geq \gamma \frac{\mu}{(1 + \mu\kappa_g)} \|\nabla f(x^k)\| = \omega \|\nabla f(x^k)\|,$$

where  $\omega := \frac{1}{2} \frac{\mu}{(1 + \mu\kappa_g)} > 0$  is introduced to ease notation. Supposing that  $\|\tilde{g}^k\| \geq \beta$  and applying the error bounds generated from fully linear models, we have

$$\|\nabla f(x^k)\| \geq \|\tilde{g}^k\| - \|\nabla f(x^k) - \tilde{g}^k\| \geq \beta - \kappa_g \Delta^k.$$

Substituting this into Equation (11.13), we find  $\Delta^k \geq \omega \|\nabla f(x^k)\| \geq \omega(\beta - \kappa_g \Delta^k)$  which may be rewritten as

$$\Delta^k \geq \frac{\omega\beta}{1 + \omega\kappa_g}.$$

Hence, if  $\tilde{g}^k$  is bounded below by  $\beta$ , then Step 1 cannot allow  $\Delta^k \rightarrow 0$ .

Consider now Step 4. Following a similar logic, but using Theorem 11.4, we know that once Step 4 is passed,

$$\Delta^{k+1} \geq \gamma \min \left\{ \frac{\|\tilde{g}^k\|}{\kappa_H}, \frac{\mu_c \|\tilde{g}^k\|}{4\kappa_f} (1 - \eta) \right\}$$

Hence, if  $\tilde{g}^k$  is bounded below by  $\beta$ , then Step 4 cannot allow  $\Delta^k \rightarrow 0$ .

Thus, if there exist  $\beta > 0$  such that  $\|\tilde{g}^k\| \geq \beta$  for every  $k$ , then  $\Delta^k$  cannot converge to 0, which contradicts Equation (11.12). Thus, we must have

$$\liminf_{k \rightarrow \infty} \|\tilde{g}^k\| = 0.$$

Finally, we apply the triangle inequality, Definition 9.2, and  $\Delta^k \leq \mu\|\tilde{g}^k\|$  (by Step 1), to see

$$\begin{aligned} 0 &\leq \|\nabla f(x^k)\| \leq \|\nabla f(x^k) - \tilde{g}^k\| + \|\tilde{g}^k\| \\ &\leq \kappa_g \Delta^k + \|\tilde{g}^k\| \\ &\leq \kappa_g \mu \|\tilde{g}^k\| + \|\tilde{g}^k\|. \end{aligned}$$

Thus,  $\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0$ . □

## 11.5. More Experiments with the Rheology Optimization Problem

We end this chapter by applying the MBTR algorithm to the rheology parameter fit optimization problem from Section 1.3.3. As with Chapter 10, since we are working with model-based methods, we focus on the smooth rheology parameter fitting problem.

**EXAMPLE 11.3.** Consider the smooth rheology parameter fitting problem from Section 1.3.3, as it was reformulated in Example 3.1:

$$\min_{x \in \mathbb{R}^3} \{\check{f}(x) : x \in [\ell, u]\}.$$

We use the 7 starting points from Example 3.5, whose function values are listed in Example 10.1.

To apply a MBTR algorithm, we generate the quadratic models in two different ways. First, we use a minimum Frobenius norm model function, using  $2n+1 = 7$  points (Definition 9.11). Second, we use a quadratic interpolation function using  $\frac{1}{2}(n+1)(n+2) - 1 = 10$  interpolation points (Definition 9.9). We set  $\Delta^0 = 1$ ,  $\mu = 0.1$ ,  $\eta = 0.05$ , and  $\gamma = 0.5$  and ran these two algorithms from each of the seven starting points, with an additional budget of 125 and 375 function evaluations. Table 11.1 presents the results. For comparison, column “CS best” in Table 10.1 presents the best result found using the CS algorithm as detailed in Example 3.5. The final columns in Table 11.1 show the results of a gradient-based trust region algorithm that uses a BFGS update to approximate the Hessian.<sup>2</sup> As the gradient-based TR converges quite quickly, instead of showing the results of 125 and 375 function evaluations, the final column states how many function evaluations were required to achieve the exact minimum (within  $10^{-6}$ ).

As in Example 10.1, using 375 function evaluations outperforms 125 additional function evaluations in every case. Comparing the 7-point MBTR and the 10-point MBTR does not present a clear winner. This is actually quite normal, as the increased accuracy in the 10-point MBTR comes at the cost of extra function evaluations.

---

<sup>2</sup>The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is a classic method that uses gradient information to iteratively update a Hessian approximation. Information can be found in most Numerical Optimization textbooks, for example [133, Chapter 6].

TABLE 11.1. Best function value  $\check{f}$  on the smooth rheology problem, generated using different initial points and two versions of the MBTR algorithm. In addition, the number of function calls required to find the solution (171.8) using a gradient-based trust-region algorithm.

| Initial Point | CS best                   |                           | 7-point MBTR              |                           | 10-point MBTR             |                           | Gradient-based TR |           |
|---------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|-------------------|-----------|
|               | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | 125<br>$\check{f}$ -calls | 375<br>$\check{f}$ -calls | Solution          | f/g-calls |
| $x_{GS}^0$    | 8033.51                   | 7878.70                   | 7974.2                    | 6822.9                    | 8291.8                    | 7861.4                    | 171.8             | 78        |
| $x_{LHS_1}^0$ | 5139.76                   | 1094.98                   | 6793.6                    | 1700.5                    | 19120.2                   | 4388.3                    | 171.8             | 38        |
| $x_{LHS_2}^0$ | 5819.71                   | 5310.60                   | 5920.0                    | 5800.0                    | 5922.4                    | 5512.7                    | 171.8             | 60        |
| $x_{LHS_3}^0$ | 4657.93                   | 4325.62                   | 4371.3                    | 4153.7                    | 4387.6                    | 4040.4                    | 171.8             | 87        |
| $x_{LHS_4}^0$ | 3752.66                   | 3218.25                   | 821.4                     | 176.8                     | 1039.7                    | 174.3                     | 171.8             | 47        |
| $x_{LHS_5}^0$ | 247.21                    | 217.62                    | 2531.9                    | 1511.2                    | 1730.4                    | 173.8                     | 171.8             | 31        |
| $x_{LHS_6}^0$ | 3691.67                   | 3382.16                   | 3762.1                    | 2202.6                    | 4983.8                    | 2276.4                    | 171.8             | 50        |

Also, in general the MBTR algorithm outperforms the CS algorithm, although several exceptions exist. Most notably the 10-point MBTR does quite poorly when starting from  $x_{LHS_1}^0$ . This is fairly typical of a MBTR algorithm, as good convergence relies on the solution point lying close to (or preferably within) the trust region. When the solution point lies far from the trust region, then a MBTR algorithm can be quite slow until the trust region moves closer to the solution point.

Comparing Table 10.1 and Table 11.1 emphasises the challenge in designing good model-based DFO methods. The MBTR algorithm outperforms the MBD algorithm quite significantly in some cases, while in other cases the opposite is true. The creation of an approximate Hessian in the MBTR comes at the cost of extra function evaluations, thus resulting in fewer iterations, making the outcome unclear.

Finally, as in Example 10.1, Table 11.1 shows the dramatic power of a gradient-based methods over the DFO methods. Even in the worst cases, the gradient-based TR converges in just 87 additional function evaluations. As mentioned many times, if accurate gradients are readily available, then DFO methods should not be used.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 11.1.** Fill in the missing details in Examples 11.1 and 11.2.

**EXERCISE 11.2.** Compare Step 2 of the MBD algorithm to Steps 1 and 2 of the MBTR algorithm. Explain the similarities and differences.

**EXERCISE 11.3.** Suppose  $\tilde{f}^k(x) = 2x_1x_2 + 7$ ,  $x^k = [0, 0]^\top$ , and  $\Delta^k = 2$ . Exactly solve the trust region subproblem.

**EXERCISE 11.4.** Suppose  $\tilde{f}^k(x) = x_1x_2 - 3x_3$ ,  $x^k = [1, 1, -2]^\top$ , and  $\Delta^k = 1$ . Exactly solve the trust region subproblem.

**EXERCISE 11.5.** + Suppose  $\tilde{f}^k(x) = -2x_1 + 2(x_1)^2 + 4x_1x_2 + 2(x_2)^2$ ,  $x^k = [0, 0]^\top$ , and  $\Delta^k = 1$ . Solve the trust region subproblem.

**EXERCISE 11.6.** Suppose  $\tilde{f}_\Delta$  is a quadratic (as in Equation (11.3)). Prove that the Cauchy value can be written as in Equation (11.5).

**EXERCISE 11.7.** Suppose  $\tilde{f}^k(x) = x_1x_2 - 3x_3$ ,  $x^k = [1, 1, -2]^\top$ , and  $\Delta^k = 1$ .

- a) Find the exact Cauchy point and Cauchy value.
- b) Find the approximate Cauchy value as given by Equation (11.4).
- c) Compare your results to Exercise 11.4.

**EXERCISE 11.8.** + Suppose  $\tilde{f}^k(x) = -2x_1 + 2(x_1)^2 + 4x_1x_2 + 2(x_2)^2$ ,  $x^k = [0, 0]^\top$ , and  $\Delta^k = 1$ .

- a) Find the exact Cauchy point and Cauchy value.
- b) Find the approximate Cauchy value as given by Equation (11.4).
- c) Compare your results to Exercise 11.5.

**EXERCISE 11.9.** Suppose  $\tilde{f}^k(x) = -2x_1 + 2(x_1)^2 + 4x_1x_2 + 2(x_2)^2$ ,  $x^k = [0, 0]^\top$  and  $\Delta^k = 1$ . Fix the accuracy parameter  $\mu_c = \frac{1}{2}$ . Use any method to find an approximate minimiser  $x_{\text{tmp}}^k$  to the MBTR subproblem such that

$$\tilde{f}^k(x_{\text{tmp}}^k) \leq \tilde{f}^k(x^k) + \frac{1}{2}\mu_c \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\}.$$

**EXERCISE 11.10.**  $\square$  Read Exercise 10.8. Suppose  $\tilde{f}^k(x) = x_1x_2 - 3x_3$ ,  $x^k = [1, 1, -2]^\top$ , and  $\Delta^k = 1$ . Fix the accuracy parameter  $\mu_c = 0.95$ . Use a line search in the direction  $-\nabla \tilde{f}^k(x^k)$  to find an approximate minimiser  $x_{\text{tmp}}^k$  to the MBTR subproblem such that

$$\tilde{f}^k(x_{\text{tmp}}^k) \leq \tilde{f}^k(x^k) + \frac{1}{2}\mu_c \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\},$$

for the accuracy parameter values  $\mu_c \in \{0.1, 0.2, \dots, 0.9\}$ . Explain how the results differ for the different values of  $\mu_c$ .

**EXERCISE 11.11.**  $\square$  Read Exercise 10.8. Suppose  $\tilde{f}^k(x) = -2x_1 + 2(x_1)^2 + 4x_1x_2 + 2(x_2)^2$ ,  $x^k = [0, 0]^\top$ , and  $\Delta^k = 1$ . Use a line search in the direction  $-\nabla \tilde{f}^k(x^k)$  to find an approximate minimiser  $x_{\text{tmp}}^k$  to the MBTR subproblem such that

$$\tilde{f}^k(x_{\text{tmp}}^k) \leq \tilde{f}^k(x^k) + \frac{1}{2}\mu_c \|\tilde{g}^k\| \min \left\{ \frac{\|\tilde{g}^k\|}{\|H^k\|}, \Delta^k \right\},$$

for the accuracy parameter values  $\mu_c \in \{0.1, 0.2, \dots, 0.9\}$ . Explain how the results differ for the different values of  $\mu_c$ .

**EXERCISE 11.12.** + Suppose  $f(x) = e^{x^2+y^2} + 2x(y-1)$ ,  $\tilde{f}^k(x) = -2x_1 + 2(x_1)^2 + 4x_1x_2 + 2(x_2)^2$ ,  $x^k = [0, 0]^\top$ .

- a) Find the exact Cauchy point as a function of  $\Delta^k > 0$ .
- b) Evaluate  $\rho^k$  at the exact Cauchy point from question a),  $x_{\text{tmp}}^k = x_c$ , to create  $\rho^k$  as a function of  $\Delta^k$ .
- c) What values of  $\Delta^k$  will result in successful iterates?



### Chapter 11 Project: Apply MBTR to the Smooth Rheology

**Problem.** Implement the MBTR algorithm using a quadratic based class of fully linear models. Apply the MBTR algorithm on the smooth rheology problem (from Section 1.3.3). Explain how (and why) your results differ from those in Table 11.1.

# *Further Remarks on Model-Based Methods*

---

Model-based methods for DFO date back to at least the early 1970s [137, 168, 169]. The first use may be Winfield’s Ph.D. thesis entitled “Function and Functional Optimization by Interpolation in Data Tables”, where he used quadratic interpolation to minimise seven different test functions [168]. Winfield derived the  $\frac{1}{2}(n + 1)(n + 2)$  data points required for quadratic interpolation and then minimised the model over a “region of validity” which is akin to modern trust regions.

The idea of model-based DFO methods was generally considered too computationally expensive until the mid-1990s, when Powell developed rigorous analysis for a method based on linear interpolation [138]. About the same time, Conn and Toint published research on using quadratic interpolation in DFO [43]. Incidentally, Conn and Toint’s 1996 article [43] may mark the first occasion that “Derivative Free Optimization” appears in the title of a research paper. These articles sparked interest into model-based DFO methods, and since then various researchers have developed new approaches and new implementations.

Many of the pivotal implementations are the works of Powell [139, 140, 141], which are still widely used today. The theory on building models in Chapter 9 can be traced back to works by Conn, Scheinberg, and Vicente [45, 46]. The term *fully linear models* was first formally defined in the work of Wild and Shoemaker [166], although they give credit to Conn, Scheinberg, and Vicente [45] for inspiring the terminology. Simplex gradients are frequently used in direct search frameworks and some of their calculus properties are exposed by Regis [145].

More recent research has explored building models via Gaussian processes [103], radial basis functions [167], weighted regression [30], and methods for nonsmooth functions [93, 111].

The Fundamental Theorem of Calculus in  $\mathbb{R}^n$  (Lemma 9.3) can be found in most multivariate calculus textbooks.

Step 2b) in the MBD algorithm is written in a simple form that sets  $\Delta^{k+1} \leq \frac{1}{2}\Delta^k$  when model inaccuracy is detected. In practice, it can be valuable to decrease  $\Delta^{k+1}$  aggressively, at least in some circumstances. One popular update rule, based on Equation (10.3), is to set

$$\Delta^{k+1} = \min \left\{ \frac{1}{2}\Delta^k, \frac{\mu^k}{1 + \mu^k} \|\tilde{g}^k\| \right\}.$$

This rule is based on approximating  $\bar{\kappa}_g$  with 1 and approximating the gradient  $\nabla f(x^k)$  by  $\tilde{g}^k = \tilde{g}_{\Delta^k}(x^k)$  in Equation (10.3). While neither of these approximations may be accurate, the minimisation of this approximation with  $\frac{1}{2}\Delta^k$  makes the rule safe in practice. However, it should be cautioned that while aggressive decreases in  $\Delta^{k+1}$  may help convergence, they may also lead to numerical instability in the algorithm that results in early termination.

It is worth noting that Theorem 11.5 can be strengthened to a full limit [47, Thm 10.13]; however, the details become more technical and a few extra subroutines must be added to the MBTR algorithm. Another important note is that Theorem 11.5 is presented in light of fully linear models, not fully quadratic models. If fully quadratic models are available, then it is often preferable to replace the minimal decreased based on the Cauchy step with a minimal decreased based on the “eigenstep”. This becomes more technical, but provides a better bound on the potential decrease in solving the trust region subproblem, which in turn can lead to better algorithmic performance.

Detailed presentation of trust region methods for smooth problems is found in the volume of Conn, Gould, and Toint [44].

## Extensions and Refinements

Part 5 gives a glimpse of possible extensions and refinements of the methods presented in this book.

- Chapter 12 discusses the various types of variables and constraints encountered in real-life optimization problems. The chapter also presents another approach to handle constraints.
- Chapter 13 discusses a few ways to use surrogates and approximations models within an algorithmic framework. Proper use of surrogates or models can drastically reduce the overall computational effort.
- Chapter 14 describes situations in which there is not one, but two conflicting objective functions that need to be considered.

Conclusion: current research in the field of derivative-free and blackbox optimization is extremely active. There are numerous other topics that could have been included in this book.

## *Variables and Constraints*

Throughout this book, we have considered the general problem of minimising a multivariate objective function  $f$  over a constraint set  $\Omega \subseteq \mathbb{R}^n$ . Let us now be more specific about the nature of the variables and of the constraint set. Consider the optimization problem

$$(12.1) \quad \min_{x \in X \subset \mathbb{R}^n} \{f(x) : c(x) \leq 0\},$$

where  $f : X \subseteq \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  and  $c : X \subseteq \mathbb{R}^n \mapsto (\mathbb{R} \cup \{\infty\})^m$  are functions with  $c = [c_1, c_2, \dots, c_m]^\top$ , and  $X$  is some subset of  $\mathbb{R}^n$ . Notice that the functions are allowed to return the value  $\infty$ , which is a convenient technique to represent evaluation failures. The entire feasible region is denoted by

$$\Omega = \{x \in X : c(x) \leq 0\}.$$

The set  $X$  can be used to define the space in which the variables live. For example, the set  $X$  is frequently taken as being  $\mathbb{R}^n$ , the space of continuous variables. Alternately, it could be the set of nonnegative variables  $\mathbb{R}_+^n$ . This is common, for example, when the variables represent lengths, areas, masses, or volumes, so the simulation fails to execute when provided with negative

values. We will see in Section 12.1 that there are other possibilities for  $X$  that are compatible with the optimization methods proposed in the previous chapters.

In Section 12.2, we will describe some of the different types of constraints that may occur in a BBO problem.

Constraints of the form  $c(x) \leq 0$  may be provided analytically, or may be computed by the same simulation that returns the objective function  $f$ , or may be provided by separate simulations. The later two cases imply that first-order information is not analytically available. In the final case, different simulation times may suggest using a different order when checking constraints. In any case, such constraints are singled out as being more informative than the constraints given by  $x \in X$ . In particular, they both delimit the feasible region and include knowledge about the amount by which a constraint is violated. Sections 12.3 and 12.4 provide strategies to exploit the knowledge of the amount by which a constraint is violated. The method in Section 12.4 is typically more effective than the extreme barrier from Chapter 8.

### 12.1. Types of Variables

Our target class of optimization problems may possess several different kinds of variables. Indeed, all types might occur in a single problem.

This information concerning the variables is normally coded in the set  $X$  rather than in the constraints  $c(x)$ . We will not strain for rigourous definitions of these types of variables because attempting to cover all possible, and mostly unrealistic, cases would lead to tedious pedantic statements that would add little to the reader's understanding. Instead, we begin with descriptive definitions and then follow with some illustrative examples of the boundaries between the types of variables.

- i. A variable is said to be *continuous* if the values that it may take include a nonempty interval. Thus, we think of continuous variables as being real vectors. However in practice, numerical computations by machines cannot deal with real numbers, as finite amounts of memory are used to store the numbers.
- ii. A variable is said to be *discrete* if for any value  $x$  that the variable may take, there is an open neighbourhood  $\mathcal{N}$  such that the only value that the variable may take in  $\mathcal{N}$  is  $\{x\}$ . This is consistent with the notion of a discrete subset of  $\mathbb{R}^n$ . A special kind of discrete variable are those that take their values from the set of integers  $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ . Another frequent type are Boolean variables restricted to the values 0 and 1.
- iii. A variable is said to be *periodic* with period  $p \in \mathbb{R}$  if given any value  $x$  the value  $x + p$  is equivalent for the purposes of evaluating the problem functions, i.e.,  $f(x + ip) = f(x)$  and  $c(x + ip) = c(x)$  for every  $x \in \mathbb{R}^n$  and for every  $i \in \mathbb{N}$ .

It is not uncommon for a problem to have some components of  $x$  that correspond to continuous variables, some to discrete variables, and any of these might be periodic.

With the exception of the GA algorithm, the algorithms presented in Parts 2 through 4 were designed for optimization of continuous variables. We next discuss how to adapt the algorithms from Parts 2, 3, and 4 where some variables are discrete, binary, or periodic.

### Discrete Variables

GA algorithms were originally devised for discrete variables. The fitness, selection, mutation, and crossover rules from Chapter 4 can be applied without any modifications. The first encoding rule from that chapter was for bounded integer variables.

The GPS and MADS algorithms require that the trial points be generated on a mesh. The set  $\mathbb{Z}^n$  is a mesh, and therefore, an easy way to handle integer variables is to make sure that the mesh point is composed of integers. This can be done for example by taking the matrix  $D$  to be equal to  $[I - I]$ , where  $I$  is the identity matrix, and by making sure that the mesh size parameter is a power of two. This is easily done with the default algorithmic values, and by choosing the initial poll size parameter  $\Delta^0$  to be an integer power of 2.

Binary variables can easily be handled by toggling their values in the poll step of direct-search methods. The natural stopping criteria for these direct search methods is to terminate as soon as a local optimizer on the integer mesh is found. Bounds on integer variables are handled easily by the extreme barrier approach from Section 8.1.

Model-based methods are not compatible with discrete variables, because they require to sample the functions in a small ball using properties of continuous variables.

### Periodic Variables

Periodic variables are typically used to model situations such as days of the week, hours in the day, or angles. For example, if  $x$  represents an angle, the blackbox returns exactly the same output when the input is  $x$  or  $x + 2\pi$ .

A first approach to dealing with such constraints might be to impose that variable is bound in the interval  $[\ell_i, u_i]$ . This is dangerous because it may introduce locally optimal solutions at the boundary. For example  $\hat{x} = 0$  is a local minimiser of  $f(x) = \sin(x)$  on the interval  $[0, 2\pi]$ , but is not a local minimiser when  $x$  is free.

Another approach consists in removing the bounds  $x_i \in [\ell_i, u_i]$ , but this can easily create scaling difficulties.

The preferred algorithmic approach for dealing with periodic variables consists of adding or subtracting an integer multiple of the period  $u_i - \ell_i$  so that the resulting value lies in the interval  $[\ell_i, u_i]$ . This ensures that a local descent step that moves  $x_i$  to one of its bounds will not get stuck at that bound, but instead, would “loop” the value of  $x_i$  around to the opposite bound.

Table 12.1 indicates the types of variables that can be *easily* handled by the algorithms from the present book. This is not to say that algorithms without checkmarks cannot handle variables of the given type, just that more work is required to deal with the situation.

TABLE 12.1. Variable types easily handled by classes of algorithms

| Algorithm | Continuous | Discrete | Periodic |
|-----------|------------|----------|----------|
| GA        | ✓          | ✓        |          |
| NM        | ✓          |          |          |
| GPS       | ✓          | ✓        | ✓        |
| MADS      | ✓          | ✓        | ✓        |
| MBD       | ✓          |          | ✓        |
| MBTR      | ✓          |          |          |

## 12.2. Types of Constraints

Many optimization algorithms apply the assumption that constraints are given in the form  $c(x) \leq 0$ . For these constraints, given  $x \in X$ , each  $c_j(x)$  returns a scalar that indicates a measure the level of  $x$ 's feasibility or infeasibility. It is often assumed that, even if the constraint is not satisfied, the output values  $f(x)$  and  $c(x)$  are valid. If  $f(x)$  and  $c(x)$  can be evaluated and produce trustworthy values at any  $x \in X$ , then we call the  $c(x) \leq 0$  constraints *relaxable constraints*. A common example of a relaxable constraint is a monetary budget. For example, a simulation provides information on the quality of a product, and the costs of building the product must be less than \$ $p$ . Even if the costs exceed \$ $p$ , the values returned by the simulation are valid and may be used to construct models.

Another type of constraint, usually included in the definition of the set  $X$ , is called *unrelaxable*. An unrelaxable constraint must be satisfied by any design or decision for which the other constraints or the decision criteria are to be evaluated. They may not return a number at all, they may simply return only a message saying that the constraint is satisfied or not satisfied, i.e.,  $x \in X$  or  $x \notin X$ . An algorithm's user may pose an unrelaxable constraint in order to restrict the search to a region where all interesting designs/decisions lie. There may be bounds imposed on the decision variables that the designer needs to have always satisfied – not just at the solution. For example, suppose we are tuning parameters in a large-scale computer simulation and  $x_i$  represents the probability of some simulated event occurring. If  $x_i \notin [0, 1]$ , then the simulation will fail to execute correctly and any output is meaningless. The feasible region for all such constraints defines the set  $X$ .

Finally, there is a subclass of unrelaxable constraints which are often called *hidden*. A hidden constraint is an unrelaxable constraint that *implicitly* excludes a portion of decision variable space that could be feasible with respect to all the other constraints. Specifically, optimization algorithms must cope with the situation when one tries to evaluate the objective

function or one of the constraint functions, and the evaluation fails (usually at the same cost in time as when it succeeds). When this happens, we say that the variable  $x \in X$  violates a hidden constraint.

Hidden constraints are common in BBO. One of the reasons is that there are situations where the blackbox code was written under the assumption that it would operate within some regime, but the optimization process explores regions of the feasible space in which the code is not robust. Hidden constraints can be handled using the extreme barrier function (Definition 8.3) that simply replaces the objective function value by an infinite value. Figure 12.1 shows a cross section of the objective function value of a real chemical engineering optimization problem, in terms of two variables (the other variables are fixed). The ceiling represents infeasible points, and the floor represents hidden constraints. For this specific problem, 43% of the calls to the simulation resulted in a crash of the simulation.

When there are no hidden constraints, then the range of applicable algorithms increases. For example, the model-based methods of Part 4 can be of precious help.

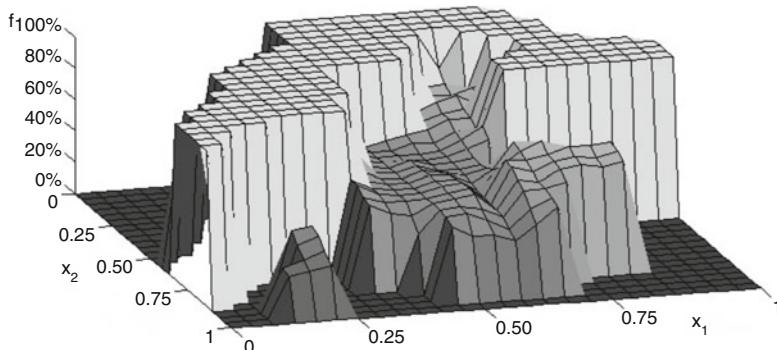


FIGURE 12.1. Objective function cross section of a chemical engineering problem

It might be tempting to think of hidden constraints as unrelaxable constraints that the user forgot to include in the problem definition. But that is far too simplistic. There are more fundamental reasons for the existence of unrelaxable constraints like the failure of a numerical method in the subroutine that evaluates one of the problem functions.

### 12.3. The Constraint Violation Function

Relaxable constraints can be violated, and the evaluation of the objective and constraint functions will succeed. The strategy described in this chapter uses the quantity by which a constraint is violated to guide the algorithm toward the feasible region. In order to achieve this goal, we borrow the constraint violation function developed in a similar context for filter methods.

**DEFINITION 12.1** (Constraint Violation Function).

The constraint violation function  $h : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  is defined as

$$h(x) := \begin{cases} \sum_{j \in J} (\max\{c_j(x), 0\})^2 & \text{if } x \in X, \\ \infty & \text{otherwise,} \end{cases}$$

where  $J = \{1, 2, \dots, m\}$  are the indices of the relaxable constraints.

With this definition,  $h : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$  is a nonnegative function, and  $x \in \Omega$  if and only if  $h(x) = 0$ . The constraint violation takes an infinite value when  $x \notin X$ , which happens when one of the unrelaxable constraint is violated, for example, when a hidden constraint is violated. In the situation where  $0 < h(x) < \infty$ , then  $x$  satisfies the unrelaxable constraints from the set  $X$  but not the relaxable ones:  $x \in X \setminus \Omega$ . For  $x \in X$ , the constraint violation function returns the sum of the square of the violations of each relaxable constraints.

**EXAMPLE 12.1.** The constraint violation function for  $\Omega = \{x \in X = \mathbb{R}^2 : x_2 \geq x_1 \geq 1\}$  is

$$h(x) = (\max\{x_1 - x_2, 0\})^2 + (\max\{1 - x_1, 0\})^2.$$

---

When defining  $h$ , one needs to make sure that the magnitude of the constraint functions  $c_j$  is comparable. A frequent strategy is to scale them so that they return infeasible values between 0 and 1.

Figure 12.2 illustrates the constraint violation function  $h$  on a problem in which  $X = \mathbb{R}_+^2$  and in which  $\Omega$  is delimited by two inequality constraints. The figure shows the sets of points for which  $h(x) = 1$  and  $h(x) = 2$ . The function satisfies  $h(x) = 0$  on  $\Omega$  and  $h(x) = \infty$  when  $x$  lies outside of the positive quadrant.

If  $c \in \mathcal{C}^1$ , then the constraint violation function  $h$  also belongs to  $\mathcal{C}^1$  on  $X$ . The function  $h$  could have been defined using the  $\ell_1$  norm, i.e., by taking the sum of the violations (rather than the squares), but this could introduce some nondifferentiability (see Exercise 12.3). It could also cause other difficulties, such as forcing one constraint to be satisfied at the expense of significantly increasing another.

The extreme barrier strategy from Section 8.1 can only be applied provided that the user-provided initial point  $x^0$  belongs to the constraint set  $\Omega$ . However, there are many situations in which no feasible points are known, but at least one point in the set  $X$  is known. The constraint violation function  $h$  may then be used in a two-step method to attempt the optimization of Problem (12.1). The two-phase extreme barrier method (below) first solves the minimisation of the constraint violation function  $h$  while ignoring the objective function  $f$ , and then pursues with the minimisation of the extreme barrier functions  $f_\Omega$  (Definition 8.3).

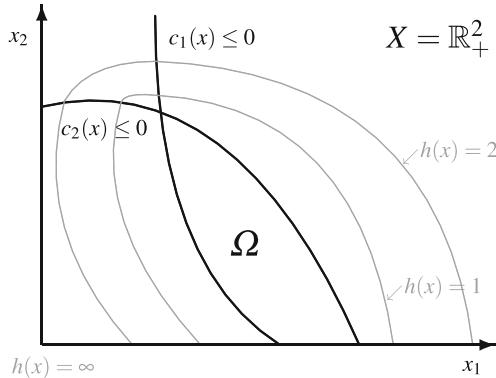


FIGURE 12.2. Illustration of the constraint violation function  $h$  on a problem in  $\mathbb{R}^2$  with two inequality constraints

**ALGORITHM 12.1.** Two-phase extreme barrier (EB)

Given  $f_\Omega : X \mapsto \mathbb{R} \cup \{\infty\}$ ,  $c : X \rightarrow \mathbb{R}^m$  and starting point  $x^0 \in X$

1. Feasibility phase

|   |
|---|
| Apply an optimization algorithm from the user-provided point $x^0 \in X$ to solve $\min_x h(x)$<br>Terminate as soon as a feasible point $\bar{x} \in \Omega$ is generated and go to 2<br>If the algorithm fails in generating a feasible point,<br>terminate by concluding that no point in $\Omega$ was found |
|---|

2. Optimization phase

|  |
|--|
| Define $f_\Omega(x) := \begin{cases} f(x) & \text{if } x \in \Omega, \\ \infty & \text{if } x \notin \Omega. \end{cases}$<br>Apply the optimization algorithm from the starting point $\bar{x} \in \Omega$ to solve $\min_x f_\Omega(x)$ |
|--|

In the feasibility phase, Algorithm 12.1 solves the unconstrained minimisation of the function  $h$ . Notice that  $h(x) = \infty$  when  $x \notin X$ , and therefore the first phase is equivalent to applying the extreme barrier strategy to the aggregation function  $\sum_{j \in J} (\max\{c_j(x), 0\})^2$  on the domain  $X$ . Furthermore, since the function  $h$  is nonnegative, we know that as soon as a trial point  $\bar{x}$  with  $h(\bar{x}) = 0$  is generated, then the optimization problem is solved to global optimality. The first phase then successfully terminates with  $\bar{x} \in \Omega$ . Otherwise, the first phase fails to produce a feasible point, and the algorithm terminates unsuccessful.

A drawback to this two-phase method is that the feasible point  $\bar{x}$  generated by the first phase does not pay any attention to the objective function value  $f$ . Consequently, it is possible that the feasibility phase generates a feasible point  $\bar{x}$  that is far from an optimal solution. Furthermore, the algorithm does not use any of the information provided by the constraint function values in the second phase. This might result in rapidly generating a feasible point with a large objective function value, and then the algorithm might spend an important computational effort in improving that solution.

Another popular method for handling constraints consists of minimising the penalised objective function  $f(x) + \mu h(x)$  for increasing values of the parameter  $\mu \in \mathbb{R}_+$ . The rationale behind this approach is that for large values of the penalty parameter  $\mu$ , the algorithm will select values of  $x \in \mathbb{R}^n$  for which  $h(x) = 0$ . Sometimes this is effective in BBO/DFO, while in other cases finding sufficiently large values of  $\mu$  leads to numerical instability.

#### 12.4. Relaxable Constraints by the Progressive Barrier

The two-phase EB algorithm from the previous section is an unsophisticated and simple way to handle constraints. We now propose the *progressive barrier* algorithm (PB) to handle relaxable constraints in a different way for iterative optimization algorithms. Again, we target Problem (12.1) in which the constraints are partitioned into the set  $X$  and the functions  $c(x) \leq 0$ .

The PB algorithm introduces a nonnegative barrier threshold  $h_{\max}^k \in \mathbb{R}_+$ , updated at the end of each iteration (denoted by the superscript  $k$ ). Any trial point whose constraint violation function value exceeds  $h_{\max}^k$  is rejected from consideration as with the EB approach. The barrier threshold  $h_{\max}^k$  is nonincreasing with respect to the iteration number  $k$  and the initial barrier threshold is set to  $h_{\max}^0 = \infty$ . Observe that EB algorithm can be seen as the particular case in which  $h_{\max}^0 = 0$  (it follows that  $h_{\max}^k = 0$  for all  $k$  since the threshold is nonincreasing).

The method will then compare pairs of trial points to rank them. Two feasible points are easily compared using only their objective function values  $f$  since their constraint violation function values  $h$  are both equal to zero. However, in order to rank two points in  $X \setminus \Omega$ , we use the following *dominance* notion.

**DEFINITION 12.2** (Dominated Points in Constrained Optimization).

*The feasible point  $x \in \Omega$  is said to dominate  $y \in \Omega$ , denoted  $x \prec_f y$ , when  $f(x) < f(y)$ .*

*The infeasible point  $x \in X \setminus \Omega$  is said to dominate  $y \in X \setminus \Omega$ , denoted  $x \prec_h y$ , when  $f(x) \leq f(y)$  and  $h(x) \leq h(y)$  with at least one strict inequality.*

*A point  $x$  in some set  $S \subset X$  is said to be undominated if there are no  $y \in S$  that dominates  $x$ .*

The dominance relation is illustrated in the next example.

**EXAMPLE 12.2.** In Figure 12.3, each circle represents the image of the functions  $h$  and  $f$  on a total of 13 trial points in  $\mathbb{R}^n$ .

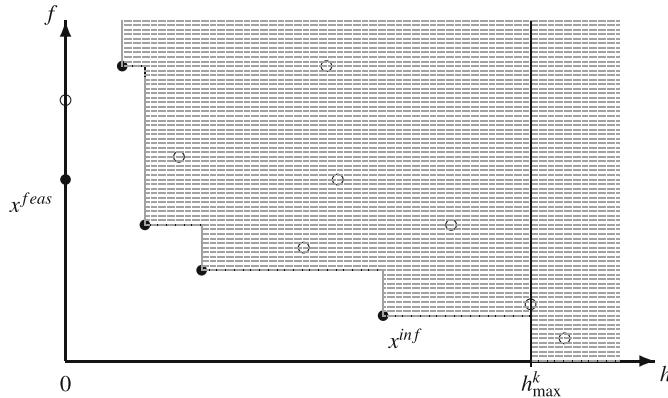


FIGURE 12.3. Undominated points are represented as solid circles, and dominated points as open circles, in the  $(h, f)$  space

In the figure, there are exactly 2 feasible points – they are located on the vertical axis where  $h = 0$ . The one with the least value of  $f$  is represented by a dark circle and denoted by  $x^{feas}$ . There are a total of 11 infeasible points, 6 of which are dominated by others. The shaded region covers all points that are dominated by some infeasible points, and the rightmost point is not dominated, but its constraint violation value exceeds the threshold  $h_{\max}^k$ . The remaining 4 undominated points appear as dark circles. The undominated infeasible solution whose constraint violation value is less than  $h_{\max}^k$  with the best objective function value is denoted by  $x^{inf}$ .

---

During iteration  $k$  of the optimization algorithm, the PB algorithm attempts to approach an optimal solution by exploring around, not one, but two incumbent solutions. The *feasible incumbent*  $x^{feas} \in \Omega$  is the feasible point with the best  $f$  value, and the *infeasible incumbent*  $x^{inf}$  is the undominated infeasible point with its  $h$  value less than  $h_{\max}^k$  with the best  $f$  value. Both incumbents are depicted in Figure 12.3. It is obvious why  $x^{feas}$  is an interesting candidate solution since it is the current best feasible solution. It is not as obvious why  $x^{inf}$  is also interesting. The reason why the PB labels it as an incumbent is that because it is the undominated one with the lowest objective function value  $f$ , it is possible that while performing some local exploration around it, and while pushing the barrier threshold parameter towards zero, that a feasible candidate point with a low objective function value is generated.

In the context of the GPS or MADS algorithm, the explorations around the incumbent solutions are simply local polls. The poll set with PB would be  $P^k = \{x + \delta^k d : x \in \{x^{feas}, x^{inf}\}, d \in D_\Delta^k\}$ . In the context of the MBD or MBTR algorithm, the exploration would rely on the construction of models around the pair of incumbent solutions.

At the end of the iteration, the iteration can be declared to be either *dominating* or *improving* (two types of success), or *unsuccessful* (failure). The possibilities are:

- i. A *dominating iteration* occurs when either a trial point  $y \in \Omega$  with  $y \prec_f x^{feas}$  or a trial point  $y \in X \setminus \Omega$  with  $y \prec_h x^{inf}$  is found. In this case, the barrier threshold is updated to  $h_{\max}^{k+1} \leftarrow h(x^{inf})$ .
- ii. A non-dominating iteration is an *improving iteration* if the algorithm has identified a trial point  $y \in X \setminus \Omega$  with  $0 < h(y) < h(x^{inf})$ . In this case, the barrier threshold is set to  $h_{\max}^{k+1} = \max\{h(v) : h(v) < h(x^{inf}), v \in V\}$ , where  $V$  is the set of all trial points at which the algorithm has computed  $f$  and  $h$  by the end of iteration  $k$ . Observe that  $h_{\max}^{k+1} = h(y)$  when  $y$  is the only point found with a better constraint violation function value than that of the incumbent.
- iii. If an iteration is neither dominating nor improving, then it is an *unsuccessful iteration*. In this case,  $h_{\max}^{k+1}$  is set to  $h(x^{inf})$ .

Algorithm 12.2 gives a high-level description of how to design a PB algorithm in a DFO context. It is left intentionally vague, as many of the algorithms presented in the book could be adapted within this framework. Notice that the starting point  $x^0$  needs to belong to  $X$ , but may be outside of  $\Omega$ . Step 2 opens up the possibility to combine heuristic methods to perform a global exploration in the space of variables. Step 3 performs a local exploration around the incumbent solutions through the use of DFO methods within a region parameterised by some radius  $\Delta^k > 0$ .

---

**ALGORITHM 12.2.** The progressive barrier algorithm (PB) \_\_\_\_\_

Given  $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ ,  $h : \mathbb{R}^n \mapsto \mathbb{R}_+ \cup \{\infty\}$  and starting point  $x^0 \in X$

0. Initialisation

|                       |                                 |
|-----------------------|---------------------------------|
| $h_{\max}^0 = \infty$ | the barrier threshold parameter |
| $\Delta^0 > 0$        | local DFO radius                |
| $k \leftarrow 0$      | iteration counter               |

1. Parameter Update

|  |
|--|
| let $x^{feas}$ and $x^{inf}$ be the two incumbent solutions<br>(there might be only one) |
|--|

2. Global search

|                   |  |
|-------------------|--|
| 3. Local DFO step | use a heuristic method to generate a finite (possibly empty) list of candidates $S^k$ in the space of variables<br>if $t \prec_f x^{feas}$ or $t \prec_h x^{inf}$ for some $t$ in $S^k$<br>increase $\Delta^{k+1} > \Delta^k$ , set $h_{\max}^{k+1} \leftarrow h(x^{inf})$ and go to 4<br>otherwise go to 3  |
| 4. Termination    | use a DFO method to generate a finite list $P^k$ of candidates near the incumbent solutions using the radius $\Delta^k$<br>if $t \prec_f x^{feas}$ or $t \prec_h x^{inf}$ for some $t \in P^k$<br>increase $\Delta^{k+1} > \Delta^k$ , set $h_{\max}^{k+1} \leftarrow h(x^{inf})$<br>else, if $0 < h(t) < h(x^{inf})$ for some previously evaluated $t \in V$<br>set $\Delta^{k+1} = \Delta^k$ and $h_{\max}^{k+1} \leftarrow \max\{h(v) : h(v) < h(x^{inf}), v \in V\}$<br>otherwise<br>decrease $\Delta^{k+1} < \Delta^k$ and set $h_{\max}^{k+1} \leftarrow h(x^{inf})$ |
|                   | stop if stopping criteria is met<br>otherwise increment $k \leftarrow k + 1$ and go to 1   |

---

The convergence analysis is beyond the scope of the present book, as it needs to be connected to the way that the global and local explorations of the DFO method are conducted. Possibilities include using direct-search in which  $\Delta^k$  would be the poll size parameter, or model-based methods in which  $\Delta^k$  would be a trust-region radius. We conclude the chapter with the following example.

**EXAMPLE 12.3.** Consider Figure 12.3, from Example 12.2, and the PB algorithm. At the beginning of iteration  $k$ , the points the points  $x^{inf}$  and  $x^{feas}$  are fixed. Suppose the remaining points at iteration  $k$ , including the global search points and local polls points, are those depicted in Figure 12.3.

Inspection of the figure reveals that the iteration is non-dominating, as no point dominates  $x^{inf}$  or  $x^{feas}$ . The iteration is improving, as a trial point exists such that  $h(y) < h(x^{inf})$ . Therefore, for the next iteration the constraint violation threshold  $h_{\max}^{k+1}$  is reduced, as illustrated in Figure 12.4. Figure 12.4 also shows that the next iteration has the same feasible incumbent solution  $x^{feas}$ , but has a new infeasible incumbent solution  $x^{inf}$ .

---

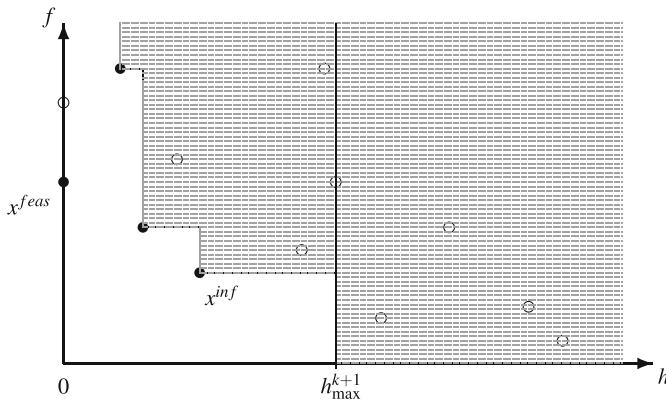


FIGURE 12.4. Incumbent solutions and constraint violation threshold at iteration  $k + 1$

### EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 12.1.** In a minimisation context, the function  $f : [0, 60] \mapsto \mathbb{R}$  was evaluated at three points:

$$f(0) = 6.0, \quad f(1) = 6.6 \quad \text{and} \quad f(59) = 8.8.$$

Knowing that  $x_1$  is a periodic variable with period  $p = 60$ , construct a quadratic interpolation polynomial and use it to propose a candidate solution in  $[0, 60]$  to evaluate  $f$ .

**EXERCISE 12.2.** + In a minimisation context, the function  $f : [0, 24) \times [0, 7] \mapsto \mathbb{R}$  was evaluated at five points:

|                   |               |               |               |               |                |
|-------------------|---------------|---------------|---------------|---------------|----------------|
| $[x_1, x_2]^\top$ | $[0, 0]^\top$ | $[0, 1]^\top$ | $[1, 0]^\top$ | $[1, 1]^\top$ | $[23, 6]^\top$ |
| $f(x_1, x_2)$     | 3.1           | 3.1           | 3.1           | 4.1           | 5.1            |

Knowing that  $x_1$  and  $x_2$  are periodic variables with periods  $p_1 = 24$  and  $p_2 = 7$ , respectively, construct a quadratic interpolation polynomial with minimal Frobenius norm and use it to propose a candidate solution in  $[0, 24) \times [0, 7]$  to evaluate  $f$ .

**EXERCISE 12.3.** Let  $X = \mathbb{R}^n$  and consider the constraint violation  $h$  from Definition 12.1.

- a) Show that if  $c \in \mathcal{C}^1$ , then  $h \in \mathcal{C}^1$ .
- b) Define  $h_1$  as

$$h_1(x) := \sum_{j \in J} \max\{c_j(x), 0\} .$$

Show that  $c \in \mathcal{C}^1$  does not imply that  $h_1 \in \mathcal{C}^1$ .

**EXERCISE 12.4.** Using the MADS algorithm, explain why it is more efficient to handle a binary variable  $x$  by setting the poll set as being the singleton  $\{1 - x\}$

- rather than treating it by adding the constraints  $x(1 - x) \leq 0$  and  $0 \leq x \leq 1$ ?
- rather than treating it as an integer variable with period 1?

**EXERCISE 12.5.** What will be the feasible point produced by the first phase of Algorithm EB with CS (use the directions  $e_1, e_2, -e_1$ , and  $-e_2$  in that specific order,  $\delta^0 = 1$  under the opportunistic strategy and the mesh size parameter is kept constant at successful iterations) with

- The starting point  $[-1, -1]^\top$  on  $\min_{x \in \mathbb{R}^2} \{x_1 + x_2 : x \geq 0\}$ .
- The starting point  $[0, 0]^\top$  on  $\min_{x \in \mathbb{R}^2} \left\{ x_1 + x_2 : \frac{1}{x_1} - x_2 \leq 0, x \geq 0 \right\}$ .

Assume a machine precision of  $10^{-15}$  (i.e., any value whose absolute value is less than or equal to the machine precision be treated as being equal to 0).

**EXERCISE 12.6.** Consider Figure 12.4. We apply the PB algorithm. The following questions are independent.

- Suppose that all search and poll points generated at iteration  $k + 1$  are dominated by  $x^{inf}$ . Trace the incumbent solutions and constraint violation threshold at iteration  $k + 2$ .
- Suppose that the first trial point visited by the search step dominates  $x^{feas}$ . Trace the incumbent solutions and constraint violation threshold at iteration  $k + 2$ .
- Suppose that the first trial point visited by the search step dominates every infeasible points generated so far by the algorithm. Trace the incumbent solutions and constraint violation threshold at iteration  $k + 2$ .

**EXERCISE 12.7.** Let  $f : \mathbb{R}_+ \mapsto \mathbb{R}$  be a monotone increasing function such that  $f(x) > e^x$ ,  $X$  be a strict subset of  $\mathbb{R}_+^2$ , and consider the optimization problem

$$\min_{x \in X} \left\{ f(x_1) : \frac{1}{10}x_1 + x_2 \geq 2, x_2 \leq 10 \right\}.$$

Suppose also that the objective function value  $f$  is known at the 10 points of the set  $A = \{[a, 1]^\top : a = 0, 1, \dots, 9\} \subset X$ .

- Write the constraint violation function  $h$  for this problem.
- Around which point of the set  $A$  would an EB algorithm explore?
- Around which point of the set  $A$  would a PB algorithm explore?
- Illustrate what the graph of  $f$  versus  $h$  might look like, and discuss which of the EB or PB algorithm would be preferable.

**EXERCISE 12.8.** + Add details to the PB algorithm so that it is coherent with the MADS algorithm. Introduce the mesh from Definition 8.1, and use a void search step. The initialisation step should be

## 0. Initialisation

|  |                                  |
|--|----------------------------------|
| $\Delta^0 \in (0, \infty)$               | initial frame size parameter     |
| $D = GZ$                                 | a positive spanning set          |
| $\tau \in (0, 1)$ , with $\tau$ rational | a mesh size adjustment parameter |
| $h_{\max}^0 = \infty$                    | the barrier threshold parameter  |
| $\epsilon_{\text{stop}} \in [0, \infty)$ | stopping tolerance               |
| $k \leftarrow 0$                         | iteration counter.               |

**EXERCISE 12.9.** + Let  $c_j : \mathbb{R}^n \rightarrow \mathbb{R}$ , for  $j \in J = \{1, 2, \dots, m\}$ , be continuously differentiable functions at a point  $\hat{x} \in \Omega = \{x \in X : c_j(x) \leq 0, j \in J\}$ , and assume that  $T_\Omega^H(\hat{x}) \neq \emptyset$ . Furthermore, assume that there is an  $\epsilon > 0$  for which for all  $x \in X \cap B_\epsilon(\hat{x})$  with  $h(x) > 0$ , there exists an  $j \in \mathcal{A}(\hat{x}) = \{j \in J : c_j(\hat{x})\}$  for which

$$c_j(x) > 0 \text{ and } \nabla c_j(\hat{x}) \neq 0.$$

Show that for every hypertangent direction  $v \in T_\Omega^H(\hat{x}) \neq \emptyset$ , there exists an  $\epsilon > 0$  for which  $h^\circ(x; v) < 0$  for all  $x \in \{x \in X \cap B_\epsilon(\hat{x}) : h(x) > 0\}$ .  
[See Section 6.5 for the definition of the hypertangent cone  $T_\Omega^H(\hat{x})$ .]



**Chapter 12 Project: The Constraint Violation Function.** Consider the set

$$\Omega = \{x \in \mathbb{R}^2 : x_2 \geq (x_1 - \pi)^2 + 1, x_2 \leq \pi, x_2 \leq x_1 + 1\}.$$

- a) Plot level sets of the constraint violation function  $h$ .
- b) Construct a simplex gradient approximation of  $\nabla h$  from

$$\mathbb{Y} = \{[0, 2]^\top, [1, 1]^\top, [1, 3]^\top\}.$$

- c) Starting at  $x^0$ , perform a line search in the direction of the negative simplex gradient to find some feasible point  $\bar{x}$  in  $\Omega$ .
- d) Suppose the computer simulation that evaluates the objective function  $f : \Omega \mapsto \mathbb{R}$  takes as input some  $x \in \Omega$  and returns a real number. However, the simulation gets caught into an infinite loop when provided with some  $x \notin \Omega$ . Propose a way to minimise  $f(x)$  over  $\Omega$ .

CHAPTER  
**13**

---

# *Optimization Using Surrogates and Models*

Section 1.4 enumerated some common features of target applications of BBO. A recurring difficulty in such applications originates from the fact that the evaluation of the objective and constraint functions is computationally expensive. In some situations, a *surrogate* optimization problem is available. That is, a problem that is considerably cheaper to evaluate and provides some insight on the true optimization problem. This chapter discusses ways of exploiting a surrogate problem as a copilot for an optimization algorithm.

## 13.1. Surrogate Problem and Surrogate Functions

Consider a constrained optimization problem of the form

$$(13.1) \quad \min_{x \in X \subset \mathbb{R}^n} \{f(x) : c(x) \leq 0\},$$

for which the objective function  $f$  and/or the constraints  $c$  are expensive to compute. Now, suppose that we have access to a second optimization problem

that takes as input the same variables and computes at a much cheaper cost a surrogate objective function  $\tilde{f}$  and surrogate constraints  $\tilde{c}$ . This creates a surrogate problem.

Let us illustrate this notion through the following example.

**EXAMPLE 13.1.** The NM algorithm described in Chapter 5 depends on four parameters: the expansion, outside and inside contraction, and shrink parameters. The most widely used values are  $\delta^e = 2$ ,  $\delta^{ic} = -\frac{1}{2}$ ,  $\delta^{oc} = \frac{1}{2}$ , and  $\gamma = \frac{1}{2}$ . These values were chosen in part for their simplicity. But could one find better values for them?

To seek better values, we could begin by selecting a large collection of test problems, say 1,000. Next, we define a function  $f$  that measures the time required by the NM algorithm to solve all 1,000 problems. The corresponding optimization problem would be to minimise  $f$  and the variables would be values of the four NM parameters. The value of  $f$  might be of the order of hours.

Notice that the overall time to solve the collection of problems is technically not a function, because solving them with the same values of  $\gamma$ ,  $\delta^e$ ,  $\delta^{ic}$ , and  $\delta^{oc}$  will likely require a slightly different amount of time.

A natural surrogate optimization problem could consist in launching the NM on a small subset of the collection of test problem. One could for example define the static surrogate  $\tilde{f}$  as the time required to solve 20 test problems. The computational time of the surrogate problem will be reduced by a factor of approximately 50 and will require minutes rather than hours.

---

How can an optimization algorithm exploit information provided by the surrogate problem to optimize the true problem? We will address this question in Section 13.2. We next formalise what is meant by surrogates.

**DEFINITION 13.1 (Surrogates).**

*The problem*

$$(13.2) \quad \min_{x \in X \subset \mathbb{R}^n} \{\tilde{f}(x) : \tilde{c}(x) \leq 0\},$$

*is said to be a surrogate for Problem (13.1) if  $\tilde{f} : X \rightarrow \mathbb{R} \cup \{\infty\}$  and  $\tilde{c} : X \rightarrow (\mathbb{R} \cup \{\infty\})^m$  share some similarities with  $f$  and  $c$  but are much faster to evaluate. The functions  $\tilde{f}$  and  $\tilde{c}$  are said to be surrogate functions of the true functions  $f$  and  $c$ .*

The definition is left intentionally vague, as the meaning of *sharing some similarities* is difficult to pinpoint. That is because a surrogate problem is not necessarily an approximation of the true optimization problem. Take the motivating example above, in which  $f(x)$  returns values whose magnitude is measured in hours, but  $\tilde{f}(x)$  returns values in minutes; however, the approximation  $\tilde{f}(x) \approx f(x)$  is very poor. Nonetheless, the surrogate shares similarities with the true problem, in particular if  $\tilde{f}(x) > \tilde{f}(y)$ , then it is

quite likely that  $f(x) > f(y)$ . The term *surrogate* rather than *approximation* was introduced to deemphasise any notion that these stand-in functions were required to be good approximations of the true functions.

**EXAMPLE 13.2.** Figure 13.1 further illustrates how a good approximation to the objective function can make bad optimization surrogate function and *vice versa*. In the figure, the objective function  $f(x)$  is noisy and is represented by the dotted curve. The dashed curve is not a good approximation of  $f(x)$  when one uses a metric returning the difference between the two functions. But the dashed curve can be used as an excellent surrogate function, since the local minima of the surrogates are very close to minimisers of  $f(x)$ . The surrogate represented by the solid curve is a considerably better approximation of the objective function  $f(x)$ , but will be a difficult surrogate to use within an optimization framework since it contains many local optima, far from the interesting zones.

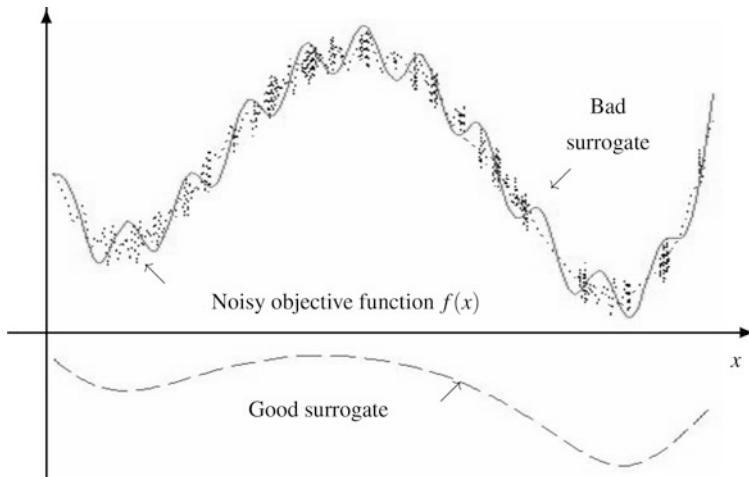


FIGURE 13.1. A good surrogate may be a bad approximation and *vice versa*

Other types of surrogates are known as simplified physics models or static surrogates, in which considerations that are difficult to handle are replaced by simpler ones. For example:

- i. Friction may be neglected in an experiment.
- ii. The internal stopping criteria within the blackbox may be relaxed for early termination.
- iii. Navier-Stokes balance equations may be replaced by simpler Euler-based ones.
- iv. A simulation that applies Monte Carlo sampling techniques may be sampled using a significantly smaller set.

- v. In fluid dynamics, finite element methods use triangular grids to analyze quantities such as displacements, stresses, and strains. The fineness of the triangular grid dictates the quality of the approximations, but greatly affects the computational cost. A coarse grid provides a natural choice for surrogates.

Surrogates may be constructed or dynamically updated. The above definition of surrogates does not exclude approximation models such as the ones presented in Part 4. One could use linear or quadratic interpolation or regression models to construct  $\tilde{f} \approx f$  and  $\tilde{c} \approx c$ . Another possibility would be to use Kriging or Gaussian process regression to construct models. These approximations can be dynamically calibrated as the optimization process is deployed, and new evaluations of the true functions are calculated.

### 13.2. The Surrogate Management Framework

Given a surrogate problem, we naturally wish to use it to improve the optimization process. One obvious approach is to locate local optimizers of the surrogate and use them as initial points for the optimization algorithm. However, this is not always the best use.

Many surrogates can be recalibrated in some manner to improve (local) accuracy. For example, a surrogate built using a model-building method (see Chapter 9) can be updated as new function values become available. Alternately, a surrogate that applies a coarse grid to analyze displacements in a fluid dynamics problem (item (v) above), can be recalibrated by applying different levels of coarseness depending on the predicted distance to optimality. In cases where the surrogates can be recalibrated, limiting the surrogates use to finding promising initial points does not employ the full power of the surrogate problem.

We now present one possible method to use a surrogate problem to improve the optimization process. The basic idea is to use the surrogate problem to rank potential trial points for the true optimization problem, and then apply an opportunistic process when applying an iteration of the algorithm on the true optimization problem. Figure 13.2 illustrates the interplay between an optimization algorithm with a pair of blackbox and surrogate optimization problems.

The optimization algorithm generates a list  $\mathcal{L}$  of tentative trial points in  $X$ . For example, the poll step in a GPS or MADS algorithm generates  $\mathcal{L} = P^k$ . In the MBD algorithm,  $\mathcal{L}$  could represent a list of potential step lengths for several different descent directions. Before sending these sets of points to the expensive true problem, the algorithm sends the entire list to the surrogate problem. The function values  $\tilde{f}$  and  $\tilde{c}$  are returned to the optimization algorithm. Based on these values, the algorithm sorts the trial points in such a way that the more promising ones are sent to the blackbox first. Finally, the true optimization problem is evaluated, and the

trial points are accepted using an opportunistic strategy. This will save the cost of launching the expensive blackbox at trial points that would likely be discarded.

If the surrogate can be recalibrated, then as the optimization proceeds the values of the true problem are used to recalibrate the surrogate problem. This is illustrated in Figure 13.2 by the arrow labelled *recalibrate*.

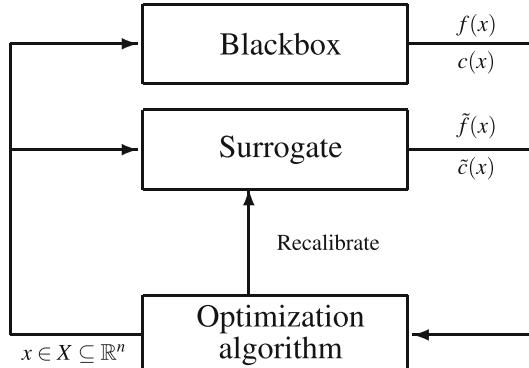


FIGURE 13.2. Interplay of an optimization algorithm with the blackbox and surrogate problems; the surrogate problem is called much more often than the blackbox problem

The surrogate management framework (SMF) in Algorithm 13.1 provides a high-level pseudo-code on this process.

**ALGORITHM 13.1. Surrogate management framework (SMF)**

Given the true functions  $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ ,  $c : \mathbb{R}^n \mapsto \mathbb{R}^m \cup \{\infty\}$  and the surrogate functions  $\tilde{f} : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ ,  $\tilde{c} : \mathbb{R}^n \mapsto \mathbb{R}^m \cup \{\infty\}$

1. Exploration using the surrogate

use the surrogate problem to generate a list  $\mathcal{L}$  of trial points  
evaluate the true problem functions at points in  $\mathcal{L}$  in an  
opportunistic way proceed to 3 if a new incumbent solution  
was generated otherwise go to 2

2. Ranking using the surrogate

use the optimization algorithm to generate a list  $\mathcal{L}$   
of trial points  
use the surrogate problem to order the points in  $\mathcal{L}$   
evaluate the true problem functions at points in  $\mathcal{L}$   
in an opportunistic way

3. Update optimization algorithm

update all algorithmic parameters and check stopping criteria  
terminate or go to 4

#### 4. Calibration of the surrogate (optional)

if possible, recalibrate the surrogate functions using  
the new function values obtained in Steps 1 and 2

---

The first step of the algorithm is flexible. The nature of the surrogate dictates the methodology that will be used to produce the list of trial points. Heuristics may be launched from different starting points. Or, in the event that the heuristic is built using quadratic models, then a dedicated quadratic solver may be used. In either case, Step 1 needs to rapidly generate a short list of candidates where the true problem is evaluated.

Step 2 performs a local exploration near the current best solution. For example, the local exploration could be constructed by the poll step of the MADS algorithm, or could result from considering different descent directions and step lengths in the MBD algorithm. In the MBTR algorithm, the trial points could be a list of points that satisfy the Cauchy point criterion discussed in Section 11.3. The objective of Step 2 is to make some local improvement to the solution.

The important distinction between Step 1 and Step 2 is in Step 2 the trial points must be generated using the processes in the optimization algorithm. This ensures that the convergence analysis for the algorithm holds true.

If Step 4 is reached, the true problem was evaluated at new trial points, but the algorithm's stopping conditions did not trigger. If possible, the new information should be used to recalibrate the surrogate problem using, for example, the model-building techniques from Chapter 9.

In the absence of constraints, ordering the points is trivially done by ranking them by increasing order of surrogate value  $\tilde{f}$ . When constraints are present, the ordering process is more elaborate. The following example describes one way to order points using the dominance relation from Definition 12.2.

**EXAMPLE 13.3.** Let  $\mathcal{L} = \{y^1, y^2, \dots, y^p\}$  be a finite list of points and let  $\tilde{h}$  denote the constraint violation function from Definition 12.1 built using the surrogate constraints  $\tilde{c}(x) \leq 0$ . The list of points  $\mathcal{L}$  is said to be ordered if given any two of its points  $x^i$  and  $x^j$ , the indices satisfy  $i < j$  whenever

- i.  $x^i$  is not dominated by any point in  $\mathcal{P}$ , but  $x^j$  is dominated by at least one point in  $\mathcal{P}$ ;
- ii.  $\tilde{h}(x^i) = \tilde{h}(x^j) = 0$  and  $x^i \prec_{\tilde{f}} x^j$ ;
- iii.  $\tilde{h}(x^i) > 0, \tilde{h}(x^j) > 0$  and  $x^i \prec_{\tilde{h}} x^j$ ;
- iv. otherwise,  $\tilde{h}(x^i) < \tilde{h}(x^j)$ .

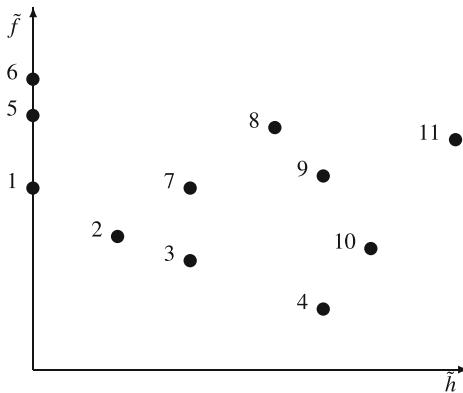


FIGURE 13.3. Ordering the trial points  $\mathcal{P} = \{y^1, y^2, \dots, y^{11}\}$  using surrogate functions

Figure 13.3 shows on the  $\hat{f}$  versus  $\hat{h}$  space how 11 elements of  $\mathcal{P}$  would be ordered using these rules.

---

We refer readers to Chapter 12 for further understanding on working with constraints in DFO.

### 13.3. Final Experiments with the Rheology Optimization Problem

Repeatedly, throughout the book we have stated that it is important to use the right tool for the job. Namely, if the optimization problem is smooth, and one has access to derivatives, then the practitioner should exploit them by using non-DFO algorithms. In the same vein, we pause to make a comment for surrogates similar to that on page 6.

If one has access to a surrogate optimization problem, then it should be exploited. Even if the surrogate values are far from the true function values, the surrogate might act as a precious copilot for the optimization algorithm, as long as they are cheap to evaluate.

These surrogates could be simplified physics models, or they could be iteratively constructed by interpolation or regression techniques. To illustrate this, we return to the nonsmooth rheology parameter fitting optimization problem from Section 1.3.3. This example was analyzed with GS, CS, GPS, and MADS in Examples 3.1, 3.5, 7.5, and 8.5.

**EXAMPLE 13.4.** We apply the MADS algorithm to the nonsmooth rheology parameter fitting optimization problem using the same seven starting points as in Example 7.5,

$$\min_{x \in \mathbb{R}^3} \{\hat{f}(x) : x \in [\ell, u]\}.$$

At each iteration, use the minimum Frobenius norm model-building technique in Section 9.5 to construct a surrogate function of the true problem. The models are calibrated by considering all previously evaluated trial points within a radius of twice the poll size parameter  $\Delta^k$  from the incumbent solution. Table 8.1 compares the solutions produced by the MADS algorithm with and without quadratic surrogate models at the 250-th, 500-th, and 1000-th function evaluations.

The value of using the surrogate problem is clear from Table 13.1. After a total of 1,000 function calls (125 to generate the initial point, and 875

TABLE 13.1. Best objective function value  $\hat{f}$  on the non-smooth rheology problem, generated by MADS without and with quadratic models from different initial points

| Initial Point | MADS                 |                      |                      | MADS with models     |                      |                      |
|---------------|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
|               | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 875 $\hat{f}$ -calls | 125 $\hat{f}$ -calls | 375 $\hat{f}$ -calls | 875 $\hat{f}$ -calls |
| $x_{GS}^0$    | 231.220              | 227.636              | 220.563              | 107.887              | 33.100               | 32.764               |
| $x_{LHS_1}^0$ | 177.506              | 38.735               | 33.992               | 43.550               | 32.738               | 32.737               |
| $x_{LHS_2}^0$ | 212.744              | 206.499              | 61.500               | 189.073              | 135.448              | 32.932               |
| $x_{LHS_3}^0$ | 183.367              | 181.538              | 80.717               | 96.844               | 34.424               | 32.726               |
| $x_{LHS_4}^0$ | 108.301              | 32.912               | 32.839               | 130.782              | 33.684               | 32.846               |
| $x_{LHS_5}^0$ | 59.654               | 37.393               | 33.275               | 39.679               | 33.595               | 33.582               |
| $x_{LHS_6}^0$ | 164.331              | 41.815               | 32.868               | 42.816               | 41.484               | 34.959               |

for the optimization process), MADS without quadratic models found good solutions ( $\hat{f}(x^{875}) < 35$ ) for four out of the seven problems. Adding the quadratic models stabilises the algorithmic performance. Indeed, the algorithm converged to a good solution from all starting points.

The appendix presents techniques to compare the performance of different algorithms. Figure 13.4 shows the accuracy profile (see Section A.3.4) for the three algorithms CS, MADS, and ‘MADS with models’. Each of the 20 test problems corresponds to the rheology problem but with a different randomly generated starting point using LHS (see Section 3.4). The accuracy value computed with Equation (A.1) uses the overall best solution produced by the 60 runs. The graph plots the proportion of problem solved as a function of  $d$ , the accuracy parameter. In order to make the graph more readable, the  $x$ -axis is in logarithmic scale.

Once again, on this example, “MADS with models” clearly dominates MADS without models, which in turns dominates CS.

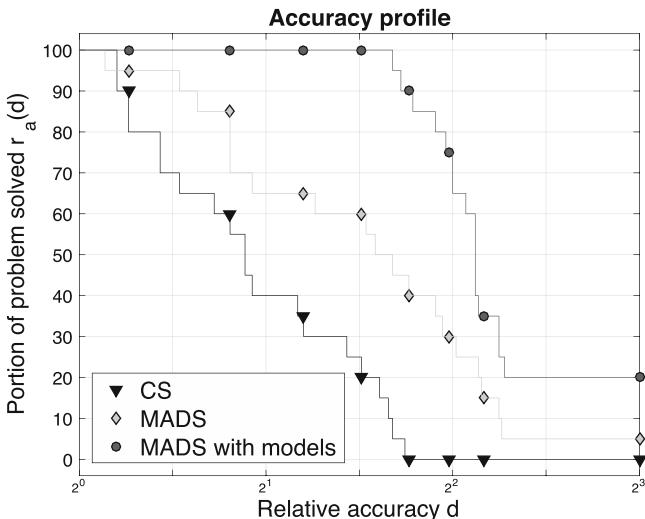


FIGURE 13.4. Accuracy profile for the rheology problem

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 13.1.** In the context of the nonsmooth rheology optimization problem from Section 1.3.3, explain why it is not a good idea to use the smooth objective function  $\tilde{g}$  as a surrogate of the nonsmooth function  $\hat{g}$ .

**EXERCISE 13.2.** We wish to solve  $\min_x \{f(x) : x \in \mathbb{R}^n\}$  where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is evaluated by the following code:

```

READ x
a ← x
δ ← ∞
WHILE δ > 10-4 DO
    b ← g(a)
    δ ← ||b - a||
    a ← b
END WHILE
RETURN f = h(a)

```

in which  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is a function that requires slightly less than 10 seconds to evaluate, and  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  is a function that evaluates very rapidly. We have observed that each evaluation of  $f$  requires between 50 and 60 minutes.

- What is the value of  $g(a)$  when the WHILE loop terminates with  $\delta = 0$ ?

- b) Propose a surrogate function  $\tilde{f}(x)$  for  $f(x)$  that requires less than a minute to evaluate.

**EXERCISE 13.3.** Suppose a computer program must be applied to perform 1,000 tasks. The time required by the computer varies from one task to another, and also depends on a set of 10 parameters, seven of which are real numbers, bounded between 0 and 100, and the three others are Booleans. The three Booleans cannot all be simultaneously equal to 1.

In order to “tune” the values of these 10 parameters, you are provided with 20 representative tasks.

- Write pseudo-code that creates a function  $f : \mathbb{R}^7 \times \{0, 1\}^3 \mapsto \mathbb{R}_+$  that takes as input the values of the 10 parameters and returns the computational time required to perform the 20 tasks.
- Give a mathematical representation of the set  $\Omega \subset \mathbb{R}^7 \times \{0, 1\}^3$  of admissible parameters.
- Write the resulting optimization problem to tune the parameters.

**EXERCISE 13.4.** Let  $\tilde{f}, \tilde{h} : \mathbb{R}^n \rightarrow \mathbb{R}$  be surrogates of the objective and constraint violation functions  $f, h : \mathbb{R}^n \rightarrow \mathbb{R}$  (see Section 12.3).

- a) Suppose that the surrogate of the objective preserves ranking:

$$\tilde{f}(x) < \tilde{f}(y) \Leftrightarrow f(x) < f(y) \quad \forall x, y \in \mathbb{R}^n.$$

Discuss the quality of this surrogate in the context of direct-search methods and model-based methods.

- b) Suppose that  $\tilde{h}(x)$  is always nonnegative and that

$$\tilde{h}(x) = 0 \Leftrightarrow x \in \Omega.$$

Discuss the quality of this surrogate.

**EXERCISE 13.5.** Propose another strategy to order the trial points from Figure 13.3 in the context where finding feasible points is difficult.

**EXERCISE 13.6.** + The function value  $f(x)$  is obtained by numerically estimating the integral

$$f(x) \approx \int_{-1}^1 g(x, u) du$$

using a Newton-Cotes rule. For a given value of  $x$ , the function  $g(x, u)$  is evaluated for each  $u \in \{-1, -0.99, -0.98, \dots, 0.99, 1\}$ . Each evaluation of  $f$  requires approximately 200 seconds.

- Using a Newton-Cotes formulae, propose a surrogate function  $\tilde{f}_N(x)$  for  $f(x)$  that requires approximately 3 seconds.
- Using a Gaussian quadrature formulae, propose a surrogate function  $\tilde{f}_G(x)$  for  $f(x)$  that requires approximately 3 seconds.

**EXERCISE 13.7.** The objective function of an optimization problem is simply

$$f(x) = \sum_{i=1}^n x_i,$$

but the constraint functions  $c(x)$  are obtained by an expensive computer simulation.

- Explain how one should build a wrapper around the simulation to be used with the MADS algorithm to avoid unnecessary evaluations of  $c(x)$ .
- Should the same wrapper be applied when considering the line-search step in the MBD algorithm?
- Explain how model functions can be used to avoid unnecessary evaluations of  $c(x)$  when the algorithm uses an opportunistic strategy.

**EXERCISE 13.8.** The objective function of an unconstrained optimization problem is

$$f(x) = \sum_{i=1}^p g_i(x)$$

where each  $g_i : \mathbb{R}^n \mapsto \mathbb{R}_+$  is computed sequentially by a blackbox simulation. The values of  $g_i(x)$  are comparable for each index  $i$ , but the computational time increases with  $i$ .

- Suggest a way to reduce as much as possible the computational time.
- In light of your answer in a), would you suggest to build a single quadratic model for  $f$ , or to build one quadratic model for each of the functions  $g_i(x)$  and to sum them?

**EXERCISE 13.9.** A multidisciplinary optimization problem evaluates the objective function value  $f(x) = g(p(x), q(x))$  by performing the following steps.

---

**ALGORITHM 13.2.** Multidisciplinary optimization problem \_\_\_\_\_

Given  $x \in \mathbb{R}^n$  and fixed parameters  $\epsilon, \epsilon_I, \epsilon_{II} > 0$ .

1. Initialisation

|                                     |                                |
|-------------------------------------|--------------------------------|
| $p^0 \leftarrow 0 \in \mathbb{R}^m$ | discipline I initial solution  |
| $q^0 \leftarrow 0 \in \mathbb{R}^m$ | discipline II initial solution |
| $k \leftarrow 0$                    | iteration counter              |

2. Discipline I

|  |
|--|
| apply Newton's method to solve the system of nonlinear equations               |
| $F_{x,p^k}(q) = 0 \in \mathbb{R}^m$ parameterised by $x$ and $p^k$ .           |
| let $q^{k+1}$ be a solution satisfying $\ F_{x,p^k}(q^{k+1})\  < \epsilon_I$ . |

3. Discipline II

|   |
|---|
| apply Euler's method with step size $\epsilon_{II}$ to solve the system of ordinary differential equations $y'_{x,q^{k+1}}(t) = \phi(t, y_{x,q^{k+1}}(t))$ with initial condition $y_{x,q^{k+1}}(0) = 0$ parameterised by $x$ and $q^{k+1}$ . |
| set $p^{k+1}$ to be the numerical solution to this differential equation.   |

4. Termination

|  |            |
|--|------------|
| if $\max\{\ p^{k+1} - p^k\ , \ q^{k+1} - q^k\ \} \leq \epsilon$ , return $f(x) = g(p^{k+1},$ | $q^{k+1})$ |
| otherwise increase $k \leftarrow k + 1$ and go to 2  |            |

---

- a) By varying the parameters  $\epsilon, \epsilon_I, \epsilon_{II} > 0$ , discuss three ways to construct surrogates of this problem.
- b) Suppose now that the parameters  $\epsilon, \epsilon_I, \epsilon_{II} > 0$  cannot be varied. Modify Step 4. to make a surrogate problem.



**Chapter 13 Project: Using the Nomad Software Package.** Find an example of BBO/DFO arising in a real-world application that may be approached using surrogates.

- a) Describe the optimization problem.
- b) Describe the surrogate and its relative computational cost.
- c) Download the Nomad software package from <https://www.gerad.ca/nomad> and experiment on this optimization problem.

CHAPTER  
**14**

---

## *Biobjective Optimization*

There are situations in which the optimization problem is driven by more than one objective function. Typically, these objectives are conflicting; for example, one may wish to maximise the solidity of a structure, while minimising its weight. In such a situation, one desires to take into account the relative tradeoffs between pairs of solutions.

The tools presented in the previous chapters may be used to identify the solutions obtained by optimizing either one of the objectives, while ignoring all others. These solutions are called the individual minima. In this chapter, we propose a way to use single-objective optimization to identify a set of tradeoff solutions to a biobjective problem. We restrict the presentation to biobjective rather than multiobjective optimization because the required computational effort increases drastically with the number of objective functions, and in the context of BBO the number of function calls needs to be controlled.

### 14.1. The Pareto Set and Front

A biobjective problem can be formally stated as

$$(14.1) \quad \min_{x \in \Omega} F(x) = (f^{(1)}(x), f^{(2)}(x))$$

where  $F : \Omega \mapsto \{\mathbb{R} \cup \{\infty\}\}^2$ , and  $\Omega \subseteq \mathbb{R}^n$  is the feasible region. However, in biobjective optimization there are usually no single vector  $x \in \Omega$  that simultaneously minimises both objective functions. The solution to Problem (14.1) consists of a set of tradeoff vectors in  $\Omega$ , called the Pareto points, and formalised shortly. The framework presented in this chapter uses any single-objective optimization algorithm to construct an approximation of this set.

In single-objective optimization, two solutions  $u$  and  $v$  in  $\Omega$  can be trivially ranked by simply comparing their objective function values  $f(u)$  and  $f(v)$ . This comparison is made more general by the following definition for the biobjective case.

**DEFINITION 14.1 (Dominated Points in Biobjective Optimization).**

Let  $u, v \in \Omega$  be two feasible points. Then, in biobjective optimization,

- $u$  is said to dominate  $v$ , denoted  $u \prec v$ , if and only if  $f^{(1)}(u) \leq f^{(1)}(v)$ ,  $f^{(2)}(u) \leq f^{(2)}(v)$ , and  $f^{(p)}(u) < f^{(p)}(v)$  for at least one index  $p \in \{1, 2\}$ .
- $u$  is said to be indifferent to  $v$ , denoted  $u \sim v$ , if and only if  $u$  does not dominate  $v$ , and  $v$  does not dominate  $u$ .

The solution set to a biobjective problem is defined through the definition of Pareto points.

**DEFINITION 14.2 (Pareto Optimality, Set, and Front).**

A point  $u \in \Omega$  is said to be Pareto optimal if and only if there is no  $w \in \Omega$  such that  $w \prec u$ . The set of Pareto optimal solutions is called the Pareto set and is denoted by  $\Omega_{\mathcal{P}}$ . It defines the solution to problem (14.1). The image of  $\Omega_{\mathcal{P}}$  under the mapping  $F$  is called the Pareto front, and is denoted by  $F_{\mathcal{P}} \subseteq \mathbb{R}^2$ .

In the biobjective case, the Pareto front is a one-dimensional curve. Any set of undominated solutions of a biobjective problem can be ordered by increasing values of the first objective and decreasing values of the second. Solving a single-objective problem consists in identifying one minimiser. Solving a biobjective problem consists in identifying the entire Pareto set. Therefore, biobjective optimization is much more difficult than single-objective optimization. Triobjective optimization is even more difficult since this ordering property is lost.

Valid lower bounds on the Pareto front values are obtained by identifying the minimal values that each of the objective function takes over  $\Omega$ . These individual minima are defined next.

**DEFINITION 14.3 (Individual Minima).**

The individual minima of  $F$  are the solutions to the single-objective optimization problems

$$\operatorname{argmin}_{x \in \Omega} f^{(p)}(x), \quad \text{for } p \in \{1, 2\}.$$

For a given objective function index  $p \in \{1, 2\}$ , if there is a unique minimiser of one individual minima problem, then the resulting solution is a Pareto point. Otherwise, it is possible that an individual minima is not a Pareto point. For example, suppose that the minimal value of  $f^{(1)}$  is zero, and minimising  $f^{(1)}$  yields a solution  $x^{(1)}$  with  $F(x^{(1)}) = (0, 2)$ . It is possible that there is another feasible solution  $u$  with  $F(u) = (0, 1.3)$ . The solution  $u$  dominates  $x^{(1)}$ :  $u \prec x^{(1)}$ .

**EXAMPLE 14.1.** The above definitions are illustrated in Figure 14.1 on a biobjective optimization problem over a tridimensional domain. The domain  $\Omega \subset \mathbb{R}^3$  is a box and illustrated in the left part of the figure.

The image  $F(\Omega)$  of the domain is the nonconvex set delimited in the right part by the curves. Four feasible points and their images are represented. One can see from the plot in the objective function space that  $x_3 \prec x_1 \prec x_2$  and  $x_4 \sim x_i$ , for  $i = 1, 2, 3$ . Dominated points are represented by open circles, and undominated ones by dark circles. The dominance zone for  $F(x_1)$  corresponds to the set of points that dominate  $x_1$  and is represented by the shaded area. Given these four points, the current approximation of the individual minima would consist of the points  $x_3$  and  $x_4$ .

The Pareto front  $F_{\mathcal{P}}$  is depicted by the union of the two thick curves in the objective space in the right part of the figure. The objective function values of the individual minima are represented by boxes. The Pareto set  $\Omega_{\mathcal{P}}$  is not represented in the figure, but would consist of a subset of the hyperrectangle to the left.

---

It is frequent in biobjective optimization that both the Pareto front and Pareto set are nonconvex or disconnected sets.

## 14.2. Single-Objective Approaches

Let ALGO denote any of the single-objective DFO optimization algorithm such as the ones presented in the previous chapters. The rest of the chapter discusses how ALGO can be used to construct an approximation of the Pareto set  $\Omega_{\mathcal{P}}$  of Problem (14.1).

The general mechanism of the biobjective algorithms presented below is that they execute ALGO on a series of single-objective subproblems. Before presenting it, let us first present two simpler ideas, and discuss their limitations.

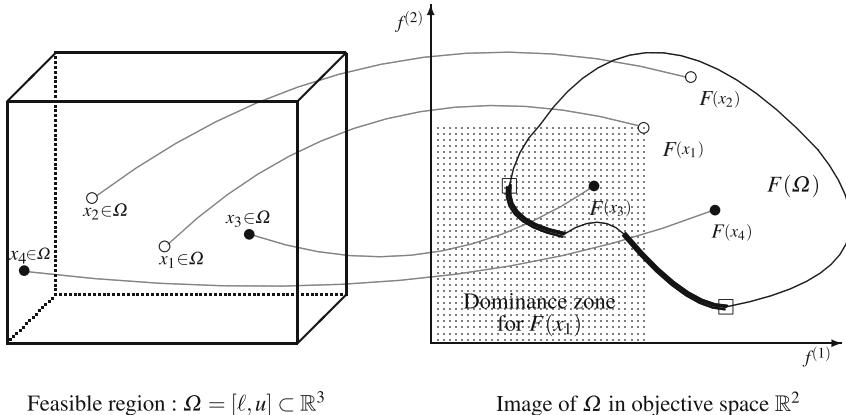


FIGURE 14.1. Pareto dominance illustrated on a biobjective problem with 3 variables

The first strategy that comes to mind consists of taking weighted linear combinations of both objectives, and to vary the weights. For any  $\lambda \in [0, 1]$ , define the single-objective optimization problem

$$(14.2) \quad B_\lambda = \min_{x \in \Omega} \lambda f^{(1)}(x) + (1 - \lambda) f^{(2)}(x).$$

Setting the parameter  $\lambda$  to 0 or 1 reduces to finding an individual minima. However, even if one would solve  $B_\lambda$  for all values of  $\lambda$  in the interval  $[0, 1]$ , this approach could fail to produce an adequate approximation of the Pareto front.

**EXAMPLE 14.2.** Figure 14.2 illustrates an example in which the Pareto front is represented by the thick curves and consists of two disjoint curves. The light diagonal line segment represents a level set of the objective function of problem  $B_\lambda$  for a given value of  $\lambda \in [0, 1]$ . The flaw with this approach is that regardless of the value of  $\lambda$ , the Pareto points in the two regions delimited by circles can never be generated.

A second idea to solve biobjective problems keeps only one of the objective function and treats the second one as a constraint with a varying threshold  $\lambda$ . For example, after having computed the individual minima  $\hat{f}^{(1)}$  for the first objective function, we define the single-objective optimization problem for any  $\lambda > \hat{f}^{(1)}$ ,

$$(14.3) \quad L_\lambda = \min_{x \in \Omega} \{f^{(2)}(x) : f^{(1)}(x) \leq \lambda\}.$$

**EXAMPLE 14.3.** Figure 14.3 illustrates again the same biobjective example. In order to construct a good approximation of the Pareto front, one would need to increment  $\lambda$  from  $\hat{f}^{(1)}$  by small step sizes. Each time an optimization problem would be solved, resulting with a point near the Pareto front.

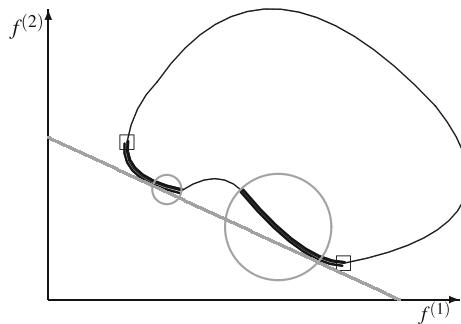


FIGURE 14.2. Taking linear combinations of the objectives fails to identify the entire Pareto front

However, notice that when  $\lambda$  takes values corresponding to the “hole” in the Pareto front, then the solution to  $L_\lambda$  will always correspond to the same point on the Pareto front.

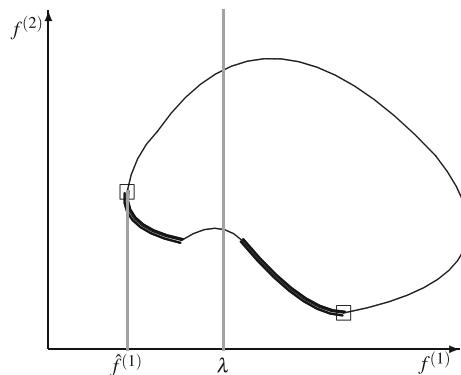


FIGURE 14.3. Treating an objective as a constraint may waste computational effort

Exercise 14.3 shows another important drawback of this approach on a trivial biobjective problem.

In the two approaches that we have mentioned so far, the level sets of the objective function are linear in the Pareto front space. Let us introduce another strategy that combines both objective functions in a nonlinear way. Each subproblem will be constructed so that ALGO generates trial points whose images lie in the dominance zone relative to a carefully chosen reference point  $r$  in the biobjective space. The dominance zone for some reference point  $r \in \mathbb{R}^2$  is the set  $\{s \in \mathbb{R}^2 : s \leq r\}$ .

The next definition proposes a function  $\phi_r : \{\mathbb{R} \cup \{+\infty\}\}^2 \mapsto \{\mathbb{R} \cup \{+\infty\}\}$  defined on the objective space, parameterised by some reference point  $r \in \mathbb{R}^2$ .

**DEFINITION 14.4** (Single-Objective Subproblem).

Let  $r \in \mathbb{R}^2$  be a reference point in the biobjective space. The single-objective optimization subproblem associated with the reference point  $r$  is defined to be

$$\min_{x \in \Omega} \phi_r(F(x)),$$

where the function  $\phi_r : \{\mathbb{R} \cup \{+\infty\}\}^2 \mapsto \{\mathbb{R} \cup \{+\infty\}\}$  is

$$\phi_r(s) := \begin{cases} -(r_1 - s_1)^2(r_2 - s_2)^2 & \text{if } s \leq r, \\ (\max\{s_1 - r_1, 0\})^2 + (\max\{s_2 - r_2, 0\})^2 & \text{otherwise.} \end{cases}$$

With this definition, if  $f^{(p)}(x)$  is infinite for one of the objective functions, for example when a hidden constraint was violated, then the value of  $\phi_r(F(x))$  is also set to  $+\infty$ . The function value  $\phi_r(F(x))$  is less than or equal to zero only at points lying in the dominance zone with respect to  $r$ . Figure 14.4 depicts level sets of the function  $\phi_r$  for a biobjective problem. ALGO will be launched on the problem from Definition 14.4. It attempts to generate trial points whose image in the objective space lie in the dominance zone with respect to  $r$ .

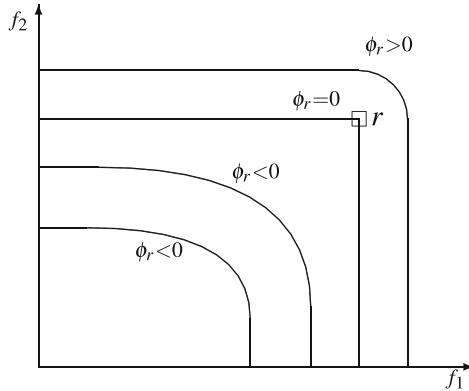


FIGURE 14.4. Level sets of  $\phi_r$  in the biobjective space

The function  $\phi_r$  satisfies the following properties:

- i.  $\phi_r(s) < 0$  if and only if  $s < r$ ,
- ii.  $\phi_r(s) > 0$  if and only if  $s_p > r_p$  for at least one index  $p \in \{1, 2\}$ ,
- iii.  $\phi_r(s) = 0$  if and only if  $s \leq r$  and  $s_p = r_p$  for at least one index  $p \in \{1, 2\}$ ,
- iv.  $\phi_r(s) < \phi_r(t)$  implies that  $s_p < t_p$  for at least one index  $p \in \{1, 2\}$ , and
- v.  $\phi_r$  is continuously differentiable.

In terms of a pair of points  $u$  and  $v$  in the constraint set  $\Omega$  with  $F(u) \neq F(v)$ , the following equivalence holds:  $u \prec v$  if and only if  $\phi_r(F(u)) \leq 0$  where  $r = F(v)$ .

The single-objective function  $\phi_r$  could have been defined in some other way. For example, it could have been defined by taking absolute values instead of squares inside the sum or the products. But this would introduce additional non-differentiability at some points, thus potentially increasing the difficulty of the problem. The next theorem validates the utilisation of single-objective subproblems.

**THEOREM 14.1.**

(Minimisers of  $\phi_r$  are Pareto) Let  $r \in \mathbb{R}^2$  be a reference point, and  $\hat{x} \in \Omega$  be a minimiser of  $\phi_r$ . If  $\phi_r(F(\hat{x})) \neq 0$ , then  $\hat{x}$  is Pareto optimal.

**PROOF.** Let  $\hat{x} \in \Omega$  be an optimal solution of a single-objective subproblem for some reference point  $r \in \mathbb{R}^2$ . By contradiction, suppose that  $\hat{x}$  is not Pareto optimal. This implies that there is a  $w \in \Omega$  such that  $w \prec \hat{x}$ , i.e.,  $F(w) \leq F(\hat{x})$ , and there is some index  $p$  in  $\{1, 2\}$  such that  $f^{(p)}(w) < f^{(p)}(\hat{x})$ .

Three cases need to be considered. First, if  $\phi_r(F(\hat{x})) < 0$ , then  $w \prec \hat{x}$  implies

$$\begin{aligned}\phi_r(F(w)) &= -\left(r_1 - f^{(1)}(w)\right)^2 \left(r_2 - f^{(2)}(w)\right)^2 \\ &< -\left(r_1 - f^{(1)}(\hat{x})\right)^2 \left(r_2 - f^{(2)}(\hat{x})\right)^2 \\ &= \phi_r(F(\hat{x})),\end{aligned}$$

contradicting the fact that  $\hat{x}$  is an optimal solution. Second, if  $\phi_r(F(\hat{x})) > 0$  and  $\phi_r(F(w)) > 0$ , then

$$\begin{aligned}\phi_r(F(w)) &= (\max\{f^{(1)}(w) - r_1, 0\})^2 + (\max\{f^{(2)}(w) - r_2, 0\})^2 \\ &< (\max\{f^{(1)}(\hat{x}) - r_1, 0\})^2 + (\max\{f^{(2)}(\hat{x}) - r_2, 0\})^2 \\ &= \phi_r(F(\hat{x})),\end{aligned}$$

contradicting again the fact that  $\hat{x}$  is an optimal solution. Finally, the same contradiction arises if  $\phi_r(F(\hat{x})) > 0$  and  $\phi_r(F(w)) \leq 0$ .  $\square$

### 14.3. Biobjective Optimization Algorithm

Let  $V \subset \mathbb{R}^n$  be a finite set of points of  $\Omega$ , and  $U \subseteq V$  be the subset of undominated points of  $V$ , i.e.,  $u$  belongs to  $U$  if and only if  $u \in V$  and there are no points  $v$  in  $V$  that dominates  $u$ . The ordering property for biobjective optimization ensures that the points of  $U$  can be ordered as  $\{u_1, u_2, \dots, u_\ell\}$  in such a way that

$$\begin{aligned}f^{(1)}(u_1) &\leq f^{(1)}(u_2) \leq \dots \leq f^{(1)}(u_\ell), \\ \text{and} \quad f^{(2)}(u_1) &\geq f^{(2)}(u_2) \geq \dots \geq f^{(2)}(u_\ell).\end{aligned}$$

The set of undominated points  $U$  may be seen as a discrete approximation of the Pareto set  $\Omega_{\mathcal{P}}$ . The first and last points,  $u_1$  and  $u_\ell$  are approximations of the individual minima of  $f^{(1)}$  and  $f^{(2)}$ , respectively.

To quantify the quality of the coverage of the undominated points in the objective function space, we introduce the following measure associated with the points of  $U$  other than the two individual minima.

**DEFINITION 14.5 (Coverage Measure).**

Let  $U$  be a finite ordered undominated set of more than two feasible points. The coverage measure of  $u_i \in U$ ,  $i \in \{2, 3, \dots, |U| - 1\}$  is defined to be

$$\delta_i = \|F(u_{i-1}) - F(u_i)\| + \|F(u_i) - F(u_{i+1})\|.$$

The coverage measure at an undominated point is the sum of the distances in the objective space to the previous and next undominated points. The larger the value of  $\delta_i$  is, the larger is the gap in the approximation to the Pareto front.

The coverage metric,  $\delta_i$ , can be used to help select a reference point for the next iteration. Once a reference point is determined, ALGO can be applied to  $\phi_r$  to further refine the Pareto front. Let us now state the biobjective algorithm. Explanatory remarks will follow.

**ALGORITHM 14.1.** Biobjective optimization algorithm (BiObj)

Given  $F : \mathbb{R}^n \mapsto \mathbb{R}^2$  and starting point  $x^0 \in \mathbb{R}^n$

0. Initialisation

|                                   |   |
|-----------------------------------|---|
| $w(x) = 1$ for all $x \in \Omega$ | implicit initialisation of the weights    |
| $n_{\text{eval}}$                 | an overall budget of function evaluations |

1. Individual minima

|  |
|--|
| apply ALGO from $x_0 \in \mathbb{R}^n$ to solve $\min_{x \in \Omega} f^{(1)}(x)$ and |
| $\min_{x \in \Omega} f^{(2)}(x)$   |

2. Reference point determination

|  |
|--|
| let $U$ be the set of feasible undominated trial points  |
| if $ U  = 1$ , set $\hat{\ell} = 1$ , and $r = (f^{(1)}(u_1), f^{(2)}(u_1))$   |
| if $ U  = 2$ , set $\hat{\ell} = 1$ , and $r = (f^{(1)}(u_2), f^{(2)}(u_1))$   |
| if $ U  \geq 3$ , choose $\hat{\ell}$ in $\operatorname{argmax} \left\{ \frac{\delta_i}{w(u_i)} : i = 1, 2, \dots,  U  \right\}$ , |
| and set $r = (f^{(1)}(u_{\hat{\ell}+1}), f^{(2)}(u_{\hat{\ell}-1}))$   |
| increase the weight $w(u_{\hat{\ell}}) \leftarrow w(u_{\hat{\ell}}) + 1$   |

3. Single-objective minimisation

|  |
|--|
| apply ALGO from $u_{\hat{\ell}}$ to solve $\min_{x \in \Omega} \phi_r(F(x))$ |
| stop if the number of blackbox calls exceeds $n_{\text{eval}}$               |

The high level overview of this approach is as follows. Given the feasible undominated points  $U$  found so far we now have 3 cases to consider based on the cardinality of the set:  $|U| = 1$ ,  $|U| = 2$ , and  $|U| \geq 3$ . If  $|U| = 1$ , then this means that there is a single undominated point  $U = \{u_1\}$ . This point will be used as a starting point in the next iteration. If  $|U| = 2$ , then the reference point is set to  $(f^{(1)}(u_2), f^{(2)}(u_1))$ . If  $|U| \geq 3$ , then the reference point is set to  $r = (f^{(1)}(u_{\hat{\ell}+1}), f^{(2)}(u_{\hat{\ell}-1}))$ , where  $\hat{\ell}$  is the index of the largest coverage measure:  $\hat{\ell} \in \operatorname{argmax}\{\delta_i\}$ . In all three cases, the logs of the ALGO runs can be used to extract a new set of feasible undominated points.

**EXAMPLE 14.4.** Figure 14.5 depicts the image in the objective space of a total of  $|V| = 13$  feasible trial points represented by circles. The dark circles represent the image of the  $|U| = 7$  undominated points, and the largest value of the coverage measure occurs at  $u_{\hat{\ell}} = u_4$ . By setting the reference point at  $r = (f_1(u_5), f_2(u_3))$  and launching ALGO from the feasible trial point  $u_4$  would result in an attempt to populate the Pareto front in the dominance zone with respect to  $r$ .

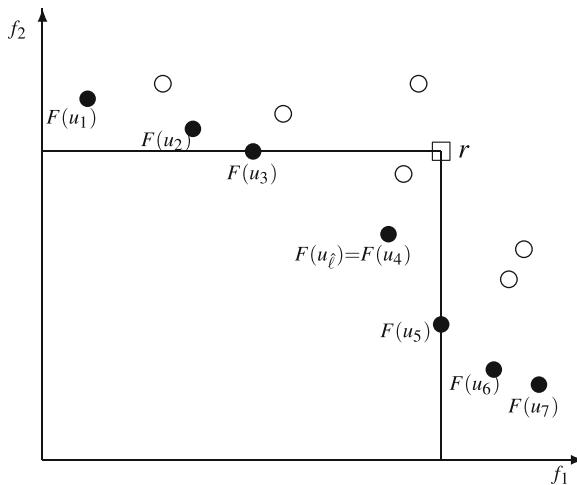


FIGURE 14.5. Construction of the reference point

---

The Pareto front is not necessarily a continuous curve, as illustrated in Figure 14.1. When this is the case, the strategy to select the trial point that has the largest coverage measure will frequently select a point near the discontinuity. In order to avoid this, a weight is taken into account into the coverage measure as follows. For every  $u_{\hat{\ell}} \in U$ , the associated weight is implicitly initialised to  $w(u_{\hat{\ell}}) \leftarrow 1$ . Then, each time that a single-objective subproblem is solved from a starting point  $u_{\hat{\ell}}$ , the weight is updated as  $w(u_{\hat{\ell}}) \leftarrow w(u_{\hat{\ell}}) + 1$ . Finally, instead of selecting the reference point by choosing the trial point

that maximises the coverage measure  $\delta_i$ , the algorithm chooses the one maximising the weighted coverage measure  $\frac{\delta_i}{w(u_i)}$ . With this strategy, each time a trial point is selected to construct a reference point, the likelihood that it will be selected again in future runs decreases.

The only important algorithmic element that we have not specified is the stopping criteria for the single-optimization runs. These are specific to the nature of ALGO. One simple possibility is that given an overall budget of  $n_{eval}$  function evaluations, to spend a fraction (such as  $\frac{1}{30}n_{eval}$  for example) on each of the single-objective minimisation.

The biobjective algorithm repeatedly solve single-objective optimization problems using ALGO. The next result shows that BiObj can inherit from the convergence analysis of ALGO. The result studies the situation where the single-objective algorithm produces a solution that satisfies the first-order optimality condition similar to that of Theorem 6.10.

**THEOREM 14.2** (Convergence of BiObj).

Let  $F \in \mathcal{C}^1$  and  $\hat{x} \in \Omega$  be an optimal solution produced by ALGO on the single-objective subproblem  $\min_{x \in \Omega} f(x)$  where  $f(x) = \phi_r(F(x))$  for some reference point  $r \in \mathbb{R}^2$ . If the single-objective optimization algorithm ensures that  $f'(\hat{x}; d) \geq 0$  for some  $d \in T_\Omega^H(\hat{x})$ , then there exists an index  $p \in \{1, 2\}$  such that  $(f^{(p)})'(\hat{x}; d) \geq 0$ .

**PROOF.** Let  $\hat{x} \in \Omega$ . We proceed by contraposition. Suppose that  $(f^{(p)})'(\hat{x}; d) < 0$  for both indices  $p = 1$  and  $p = 2$ . Therefore

$$0 > (f^{(p)})'(\hat{x}; d) = \lim_{t \searrow 0} \frac{f^{(p)}(\hat{x} + td) - f^{(p)}(\hat{x})}{t}.$$

The result follows by observing that  $f^{(p)}(\hat{x} + td) < f^{(p)}(\hat{x})$  for both indices  $p \in \{1, 2\}$  implies  $\phi_r(F(\hat{x} + td)) < \phi_r(F(\hat{x}))$ .  $\square$

The following final example studies the output produced by the biobjective algorithm on a blackbox test problem from chemical engineering.

**EXAMPLE 14.5.** Figure 14.6 illustrates the results of three applications of the biobjective optimization algorithm to a blackbox problem with 8 bound constrained variables subject to 10 blackbox constraints, 4 of which are Boolean values that simply indicate if satisfied or not. The blackbox consists of a C++ code that simulates a styrene production process and uses common methods such as Runge-Kutta, Newton, fixed points, secant, bisection, and many others. The pair of objective functions are to maximise the net present value of production and to maximise the styrene purity. while satisfying economical requirements as well as industrial and environmental regulations. Both objectives are multiplied by  $-1$  because we use a minimisation rather than a maximisation framework.

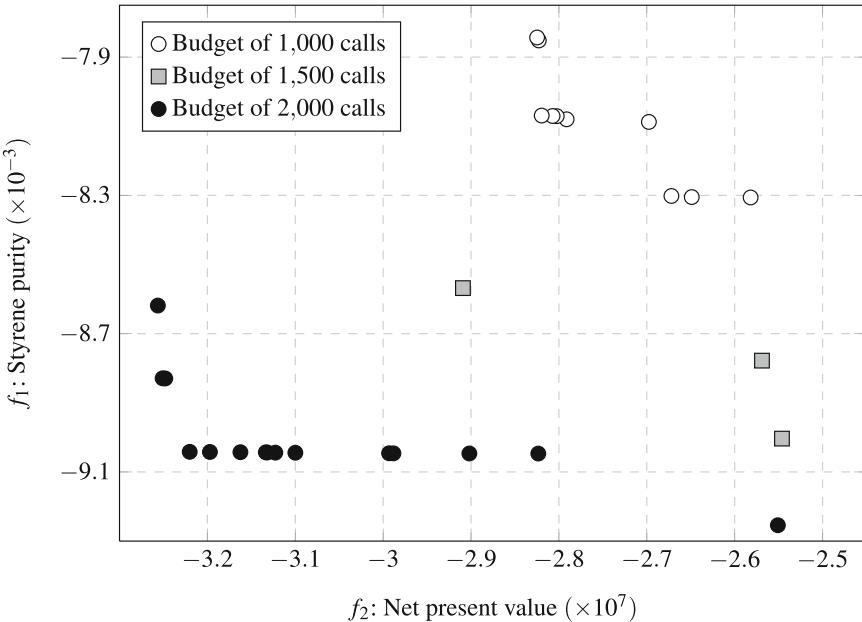


FIGURE 14.6. Successive Pareto front approximations on a styrene production problem

Each evaluation requires approximately one second to compute, and on almost 50% of the calls the simulation fails and returns an error message. When this occurs, the two outputs representing the objective function values and the 10 outputs representing the constraint function values are set to `Nan`. This fragility makes it difficult to construct approximation models using the tools presented in Part 4.

The optimization was conducted by the `Nomad` implementation of the MADS algorithm from Chapter 8. The figure only shows the undominated feasible points generated during the optimization process for three different runs. The first run is represented by the white circles and used an overall budget of  $n_{eval} = 1,000$  simulation calls. The second one is depicted by the grey squares and used a total of 1,500 simulation calls, and the third one is represented by dark circles and used 2,000 calls. The first run generated 11 undominated points and launched 11 single-objective minimisations including the two to find the individual minima of both objective functions. The second run produced only 3 undominated points using 13 single-objective minimisations. Notice that the leftmost grey square dominates all the white circles. The logs of the run reveals that the twelfth single-objective minimisation generated a point that removed 8 points that were candidates to be on the Pareto list. The third run produced 19 undominated points using 35 single-objective minimisations.

Without any surprise, the run with the largest budget of function calls produced an approximation of the Pareto front that is much better than the others. In fact all the white circles are dominated by a single grey square and all grey squares are dominated by many single black circle.

The figure also reveals tradeoffs between the two objectives. For example, it is clear from the third run that many points have a styrene purity value near  $f_1 \approx -9.1 \times 10^{-3}$  but the corresponding net present value  $f_2$  varies in the wide interval from  $-3.23 \times 10^7$  to  $-2.82 \times 10^7$ . All but one of these points could be discarded in practice.

---

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE 14.1.** Let  $u$ ,  $v$ , and  $w$  be vectors in the constraint set  $\Omega$ . Answer by TRUE or FALSE.

- a)  $u \prec u$ .
- b) It is possible that  $u \prec v$  and  $v \prec u$ .
- c)  $u \prec v \prec w$  implies that  $u \prec w$ .
- d)  $u \sim v$  if and only if the indices  $\{i, j\} = \{1, 2\}$  satisfy  $f^{(i)}(u) < f^{(i)}(v)$  and  $f^{(j)}(u) > f^{(j)}(v)$ .

**EXERCISE 14.2.** Consider the first objective function  $f^{(1)}$ .

- a) Show that when there is a unique individual minima, then it is a Pareto point.
- b) Give a biobjective example where an individual minima is not a Pareto point.

**EXERCISE 14.3.** Consider the trivial biobjective problem in which  $F(x) = (x_1, x_2)$  and  $\Omega = \mathbb{R}_+^2$ . Suppose that the individual minimal found by minimising  $f^{(1)}(x)$  is  $[0, 1]^\top$ , and that the solution to  $L_\lambda$  (Equation (14.3)) for  $\lambda > 0$  is  $[\lambda, 0]^\top$ .

- a) Show that the Pareto set is  $\Omega_{\mathcal{P}} = \{[0, 0]^\top\}$ .
- b) Show that the approach involving  $L_\lambda$  never generates a Pareto point.

**EXERCISE 14.4.** Show that the function  $\phi_r$  satisfies the following properties.

- a)  $\phi_r(s) < 0$  if and only if  $s < r$ .
- b)  $\phi_r(s) > 0$  if and only if  $s_p > r_p$  for at least one index  $p \in \{1, 2\}$ .
- c)  $\phi_r(s) = 0$  if and only if  $s \leq r$  and  $s_p = r_p$  for at least one index  $p \in \{1, 2\}$ .
- d)  $\phi_r(s) < \phi_r(t)$  implies that  $s_p < t_p$  for at least one index  $p \in \{1, 2\}$ .
- e)  $\phi_r$  is continuously differentiable.

**EXERCISE 14.5.** Let  $u$  and  $v$  be points in the constraint set  $\Omega$  with  $F(u) \neq F(v)$ . Show that  $u \prec v$  if and only if  $\phi_r(F(u)) \leq 0$  where  $r = F(v)$ .

**EXERCISE 14.6.** Let  $y \in \Omega$  be a feasible point, and  $r = F(y) \in \mathbb{R}^2$  be a reference point in the biobjective space.

- What is the value of  $\phi_r(F(y))$ ?
- Consider  $u \in \Omega$ . What is the value of  $\phi_r(F(u))$  if  $u \prec y$  and  $f^{(p)}(u) = f^{(p)}(y)$  for some  $p$  in  $\{1, 2\}$ ?
- Show that  $\phi_r(F(x)) = \text{dist}^2(F(x), D)$ , the distance (see Definition 2.8 with the  $\ell_2$  norm) in the objective space from the image of some  $x \in \Omega$  to the set  $D = \{s \in \mathbb{R}^2 : s \leq F(y)\}$ .

**EXERCISE 14.7.** Show that when  $|U| \geq 3$ , the starting point  $u_{\hat{\ell}}$  used by the biobjective optimization algorithm satisfies  $\phi_r(F(u_{\hat{\ell}})) < 0$  and  $\phi_r(F(u_{\hat{\ell}-1})) = 0 = \phi_r(F(u_{\hat{\ell}+1}))$ .

**EXERCISE 14.8.** Consider the biobjective problem with two variables

$$\min_{x \in \mathbb{R}^2} \{F(x) = (3x_1, 2\sqrt{x_2}) : x_1 + x_2 \geq 1, x \geq 0\}.$$

- What is the Pareto set?
- What is the Pareto front?
- Show that it is possible to choose  $x^1$  and  $x^2$  in the set of individual minima of both objective functions such that the line segment joining  $F(x^1)$  and  $F(x^2)$  is disjoint from the Pareto front.

**EXERCISE 14.9.** + Let  $F$  be Lipschitz near some  $\hat{x} \in \Omega$  with  $F(\hat{x}) < r \in \mathbb{R}^p$ , and let  $d \in T_{\Omega}^H(\hat{x})$  be such that  $(f^{(p)})^\circ(\hat{x}; d) < 0$  for both  $p$  in  $\{1, 2\}$ .

- Show that  $\phi_r^\circ(\hat{x}; d) < 0$ .
- Show that  $\nabla \phi_r(\hat{x})^\top d < 0$  if  $\top$  is strictly differentiable near  $\hat{x}$ .

□

**Chapter 14 Project: Tuning Runge-Kutta on Two Families of ODE.** We revisit the project of Chapter 3 from page 51 on tuning Runge-Kutta parameters. But this time we would like to find parameters that are adequate for two families of ODE.

In addition to the system of ODE from the previous project (which will now be referred to as family  $\mathcal{A}$ ), we consider the family  $\mathcal{B}$ :

$$\mathcal{B} : \left\{ \begin{array}{lcl} y'_i(t) & = & \left(\frac{y_i}{t}\right)^2 + \frac{i(i+1)}{2y_{i+1}} - \frac{i^2}{t}, & i = 1, 2, \dots, \ell-1 \\ y'_\ell(t) & = & \left(\frac{y_\ell}{t}\right)^2 + \frac{\ell}{2y_1} - \frac{\ell^2}{t}, \\ \text{with } y_i(1) & = & i, & i = 1, 2, \dots, \ell. \end{array} \right.$$

One again, we wish to tune the parameters  $\beta_1$  and  $\beta_5$ , but this time on both systems. In both families we will consider systems of  $\ell \in \{4, 5, 6\}$  equations to estimate the value of  $y(4) \in \mathbb{R}^\ell$  using a number of steps  $n$  ranging between 145 and 150.

Implement the BiObj algorithm using the DFO algorithm of your choice, and use it to plot an approximation of the Pareto front, where  $f_1$  is an error measure associated with family  $\mathcal{A}$  and  $f_2$  is an error measure associated with family  $\mathcal{B}$ . In the plot, include the points corresponding to the classical Runge-Kutta values  $\beta_1 = \frac{1}{2}$  and  $\beta_5 = 0$ , Ralston's values  $\beta_1 = \frac{2}{5}$  and  $\beta_5 = -\frac{1523\sqrt{5}}{1276} - \frac{975}{2552}$ , as well as Gill's values  $\beta_1 = \frac{1}{2}$  and  $\beta_5 = \frac{-1}{\sqrt{2}}$ .

# *Final Remarks on DFO/BBO*

---

The final part of the book contained some extensions and refinements of DFO/BBO algorithms. Specifically, we selected to cover types of variables and constraints, utilisation of surrogates, and biobjective problems.

The term “hidden constraint” was introduced in [40] and further studied in [39]. Experiments on the design of an helicopter rotor failed due to hidden constraints on 60% of the simulation calls [33, 34]. Experiments on the treatment of spent potliners [18] in the aluminium industry failed on 43% of the calls to the simulation (Figure 12.1). Recently, Le Digabel and Wild [113] presented a taxonomy to distinguish the numerous types of constraints in simulation-based optimization. They provide formal definitions of quantifiable, relaxable, *a priori*, and hidden constraints.

Other avenues exist for the treatment of nonlinear constraints in the context of BBO. The progressive barrier presented in Chapter 12 was presented in [17], which evolved from the filter approach [14] inspired by the work of Fletcher et al. [74, 73, 75]. A hybrid version of the extreme and progressive barrier approaches is defined in [22], and called the “progressive-to-extreme barrier”. Techniques for dealing with equality constraints over manifolds and linear equalities are studied in [65, 66, 67] and in [24, 88, 124, 152], respectively.

The surrogate management framework from Section 13.2 was introduced in [35, 59] to ensure that the optimization process converges to a solution of the original target problem. Variable surrogate precision is exploited in [135] to reduce the overall computational effort. Ordering trial points in the presence of constraints is proposed in [42]. Model-based surrogates include quadratic models [42, 52], DACE Kriging [32, 50, 121, 154], treed Gaussian processes [86], and radial basis functions [31, 134, 149, 166]. Surrogate models for mixed-integer blackbox problems are presented in [129, 130].

The biobjective algorithm presented in Chapter 14 first appeared in [19], and is inspired by the “Normal-boundary intersection” approach for nonlinear programming [56]. Other progress in bi- and multiobjective DFO can be found in [21, 53, 146, 147].

Research is currently very active in the DFO/BBO area, and we had to make some difficult editorial choices in what to present and what to leave out. For example, we could have presented worst case analyses for direct search methods [62, 81, 87, 164] or for DFO trust region methods [82]. We could also have adapted some of the direct search methods to identify globally optimal solutions [51, 70, 79, 119], or discussed some of the new model-based methods for nonsmooth optimization problems [92, 93, 91, 111]. Finally, we could have examined research focusing on mixed-integer [120, 136] and categorical variables [1, 4, 12, 108, 123] for BBO.

# *Comparing Optimization Methods*

This book presented many optimization algorithms specifically designed for blackbox optimization, specifically, for blackboxes that provide function values, but no first-order (gradient) or second-order (Hessian) information. At this point, a natural question arises: “*which method should we use?*”.

Before turning to this question, let us make a few comments. First, if first-order information is available, then an algorithm that makes use of it will (almost) always outperform a derivative-free method. Second, the algorithms presented in this book do not include all of the most advanced methods in derivative-free optimization (DFO). Rather, they are a selection of the foundational algorithms of DFO. It is our hope that understanding the methods in this book will provide the necessary background to study, research, and apply modern DFO.

Returning to the appendix’s principal question “*which method should we use?*”, it should be clear that the algorithm that is best on one problem may not be the best on a different problem. While there will certainly never be a universal winner, there can be great value in comparing two or more methods and trying to determine which method might “work best” for a given set of problems. If the *test set* of problems is representative of a particular application, then it is reasonable to assume that the “best” method for the

test set may, in general, be effective for that application. Therefore, in this appendix we provide an overview of good practice for comparing optimization methods. While we focus on techniques appropriate for DFO, many of the techniques are appropriate for comparing optimization algorithms in general.

The process of comparing optimization methods can be separated into three parts:

- i. Select a test set.
- ii. Collect the data.
- iii. Analyze the results.

While most of the work lies in analyzing the results, in order to create a useful comparison, it is important to perform all three of these steps well.

The contents of this appendix are derived from the benchmarking work of Dolan and Moré [63] and of Moré and Wild [128], and the survey paper [27].

### A.1. Test Sets

When comparing two optimization methods, it is obviously important to run both methods on the same optimization problem, starting with the same initial conditions. A *test set* is a collection of *test problems*.

There are several academic test sets that are generally accepted as reasonable starting points for comparing optimization problems. However, if the goal is to determine which method to use for a real-world application, then it is much better if the test problems are representative of that real-world application. In any case, the test set should contain enough test problems to ensure the results are reliable and not subject to random errors. A reasonable rule-of-thumb is to aim for test sets of *at least* 20 problems. Of course, more is preferred. One way to increase that number is to define a test problem by its blackbox and by its initial point. For example, running the same blackbox from 10 different starting points defines 10 distinct test problems. This strategy is only pertinent for algorithms that require an initial point. Moreover, this approach should be used with caution, as any hidden structures within the test problem will be reproduced in all variants of the problem.

An obvious rule, that is often neglected in practice, is that all algorithms should be started using the exact same initial information. For example, consider a comparison between two algorithms, where the first uses a Latin hypercube sample (LHS) to select an initial point, and the second simply starts at a single input point. If the work done to create the LHS is not considered when comparing the results, then the first algorithm will appear stronger than it actually is. Even if the work is counted, if the starting point for the second algorithm is a local minimiser, then the second algorithm will appear weaker than it actually is (as it will have a hard time breaking free of the local minimiser).

In many cases, comparisons of optimization algorithms use either academic test sets or artificially generated test sets. In these cases, it is possible to accidentally bias the results in several other manners. For example, recall

Example 3.3 where CS fails to minimise  $f(x) = \max\{|x_1|, |x_2|\}$  due to the poor choice of starting point  $x^0 = [1, 1]^\top$ . For almost any other starting point, CS will successfully minimise this convex function.

Artificial test sets can also have more subtle hidden structures that are not realistic in real-world applications. For example, many artificially generated problems have their solution at the origin, and integer-valued initial points. This can lead to the CS algorithm appearing stronger than it really is. One quick test to check for hidden biases in an algorithm/test set is to minimise  $f(x)$  starting at  $x^0$ , and then select two random vectors  $p \approx q$  and minimise  $\hat{f}(x) = f(x + p)$  starting at  $\hat{x}^0 = x^0 + q$ . The results of both tests should be similar; if they are not, then some hidden structure is likely being exploited.

A summary of our basic rules to selecting a good test set follows.

- i. **Include many problems.** The more problems that a test set contains, the more reliable the results of the experiment are.
- ii. **Represent the application.** Algorithms that perform very poorly for certain styles of problems may perform very well for other problems. Whenever possible, select a test set that is representative of the end application.
- iii. **Avoid biased initial conditions.** All algorithms should be started using the same information.
- iv. **Avoid hidden structures.** Carefully examine test sets to ensure no hidden structure is helping or hindering certain algorithms.

One of the most popular locations to find academic test problems is the CUTER/est Test Set [158].

## A.2. Data Collection

Given a test set and a collection of optimization problems, the next step is to run each algorithm on each test problem. Each test should be given the same computational hardware, i.e., run on the same computer. If timing results are to be compared, then each algorithm should be coded in the same language with a similar quality of implementation. However, in DFO, it is more common to compare the number of blackbox evaluations, so this condition becomes less critical.

In comparing optimization algorithms, the main goal is to compare the *efficiency* and the *quality of the solution*. Efficiency refers to the computational effort required to obtain a solution. Quality of the solution refers to the precision of the algorithm's final output. Obviously, both of these measures are important in selecting which algorithm to use. Unfortunately, doing better in one category often means doing worse in the other.

**Efficiency.** Data focusing on efficiency include running time and the number of fundamental evaluations. Running time is usually collected using CPU time (although modern usage of parallel processors is making this decision less clear). However, in DFO, we are often more interested in the number

of fundamental evaluations. The term fundamental evaluation is used to refer to algorithm calls to the blackbox that provided fundamental information about the optimization problem. In DFO, this means number of function evaluations (often referred to as  $f$ -calls).

The reason DFO is more interested in the number of  $f$ -calls than the running time is that in DFO it is assumed that a function call for a real-world problem will be the most time-consuming portion of the entire optimization process. This gives comparing DFO algorithms a certain advantage, as we can use test problems where function evaluations are very fast, and generalise some results to real-world situations where function evaluations are very slow. However, keep in mind that the DFO algorithm's behaviour may differ on the test and real problems.

**For the remainder of this appendix, we assume that efficiency is measured in terms of number of function evaluations.** This is only to make presentation cleaner, and it should generally be obvious how to adapt this assumption if another measure of efficiency is desired.

**Accuracy of the Solution.** The other important measure when comparing optimization algorithms is the accuracy of the solution. If a good solution is available (as is often the case with artificial test problems), then the quality of a solution can be gauged using the following formulae:

$$(A.1) \quad f_{\text{acc}}^N = \frac{f(x^N) - f(x^0)}{f(x^*) - f(x^0)},$$

where  $x^N$  is the best point found by the algorithm after  $N$  function evaluations,  $x^0$  is the initial point, and  $x^*$  is the best known solution. We either consider unconstrained optimization problems, or constrained optimization problems for which the initial point  $x^0$  is feasible. The subscript **acc** stands for *accuracy*. The value  $f_{\text{acc}}^N$  belongs to the interval  $[0, 1]$  and gives the relative ratio of the quality of the best point found to the starting point – the relative improvement produced using  $N$  function evaluations. A ratio of  $f_{\text{acc}}^N = 0$  indicates that the algorithm was unable to improve the given initial solution and  $f_{\text{acc}}^N = 1$  indicates that the algorithm found the best known solution.

When known no good solution is available prior to the tests, then the value  $x^*$  in Equation (A.1) can be replaced by the best point found over all tested algorithms. This is often the case for real-world test problems. (Of course, if all algorithms fail to find any improvement, this results in  $x^* = x^0$ , creating division by zero errors. In this case, the test problem should probably be removed from the analysis, and be tagged as a hard problem for these algorithms.)

**Stopping: Fixed-Target Versus Fixed-Cost.** One final decision is required during the data collection stage, when to stop each algorithm. If the algorithms include the same stopping tests, then this can be accomplished by imposing the same stopping tolerances on each algorithms. However, it is

much more common that each algorithm has different stopping tests, so such a simple approach is often impractical.

Two common stopping techniques used in comparing optimization algorithms are the *fixed-target* and *fixed-cost* techniques.

For *fixed-target* we must have a known solution to each test problem. When this is available, the fixed-target method is to determine the number of function calls required for each algorithm to obtain a solution that is within a target accuracy of the known solution. For example, given target accuracy  $\varepsilon_{\text{target}}$ , find  $N$  such that  $f_{\text{acc}}^N \geq 1 - \varepsilon_{\text{target}}$ . If an algorithm is unable to achieve the fixed-target accuracy, then it is considered unsuccessful on that test problem. Note that other fixed-target tests, for example  $\|x^N - x^*\| < \varepsilon_{\text{target}}$  or  $f(x^N) - f(x^*) < \varepsilon_{\text{target}}$ , could also be used.

The *fixed-cost* approach does not require a known solution and is often simpler to implement. In the fixed-cost approach, we simply run each algorithm until a target number of function evaluations has been exhausted. One item of note in the fixed-cost approach is that many algorithms are implemented in such a way that they can only terminate at the end of an iteration, so giving a budget of  $N$  function evaluations will often result in an algorithm using slightly more than  $N$  function evaluations. As long as  $N$  is sufficiently large, this should not cause much difficulty in analysis.

### A.3. Data Analysis

Once the tests have been performed and the data collected, the next step is to analyze the results. As a first pass, some basic statistics can be examined: average solve time, average solution quality, percentage of problems successfully solved, etc. Occasionally, the basic statistics are so overwhelmingly in favor of a certain algorithm that conclusions can be drawn immediately. However, it is much more common that such statistics are insufficient to identify a clear winner. In these cases, most research will turn to more advanced data visualisation techniques.

Before we discuss data visualisation, let us point out that in order to maximise research replicability (as is required by the scientific process), when presenting algorithm comparisons it is important to allow access to the collected data. This is most easily done using numerical tables. However, such tables are generally cumbersome, so are best relegated to an appendix or online database. The data necessary to construct the plots presented in this section can be stored in a three-dimensional array, indexed by the algorithm number  $a \in \mathcal{A}$ , the problem number  $p \in \mathcal{P}$  and the function evaluation number  $N \in \mathbb{N}$ . The value stored in the array at location  $(a, p, N)$  would be the best objective function value found by algorithm  $a$  on problem  $p$  after  $N$  function evaluations.

**A.3.1. Convergence and Trajectory Plots.** *Convergence* and *trajectory plots* are specialised plots that are often used as a first visualisation of optimization algorithm performance.

In DFO, convergence plots visualise the performance of different optimization methods by plotting the best feasible objective function value found against the number of function evaluations used. Convergence plots are always nonincreasing piecewise constant functions. Each step corresponds to a different solution whose objective function value is read on the  $y$ -axis.

Trajectory plots are restricted to functions of two variables. A trajectory plot is created by plotting the contour plot of the objective function (when such information is available) and then plotting the paths that connect the points generated by each iteration of each algorithm when applied to the objective function. An example of convergence and trajectory plots is given in Figure A.1.

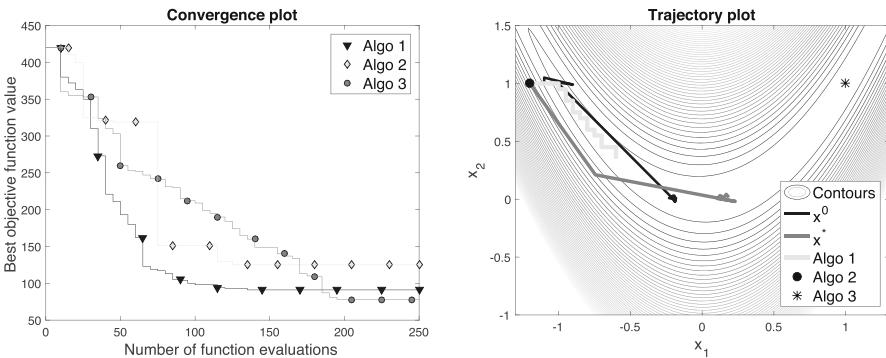


FIGURE A.1. Example of a convergence and a trajectory plot for a given test problem

While convergence and trajectory plots are useful for building an understanding of how an algorithm behaves, they are not particularly good for determining which algorithm performs the best overall. The main drawback is they can only present the results for one test problem at a time.

**A.3.2. Performance Profiles.** *Performance profiles* are currently one of the most common ways to compare several algorithms across a large test set of problems. Performance profiles are designed to capture information on efficiency (speed of convergence) and robustness (portion of problems solved) in a compact graphical format.

To define performance profiles, we consider a set of test problems  $\mathcal{P}$  and a set of optimization algorithms  $\mathcal{A}$ . Suppose each test problem is solved by each algorithm. Let  $f_{\text{acc}}^N \in [0, 1]$  be the corresponding accuracy value from Equation (A.1) at evaluation number  $N$ .

From these values, we next define Boolean variables defining a convergence test parameterised by some fixed scalar  $\tau \in [0, 1]$  called the tolerance:

$$T_{a,p} = \begin{cases} 1 & \text{if } f_{\text{acc}}^N \geq 1 - \tau \text{ for some } N, \\ 0 & \text{otherwise.} \end{cases}$$

We say that *algorithm*  $a \in \mathcal{A}$  *successfully solved problem*  $p \in \mathcal{P}$  *within tolerance*  $\tau$  when  $T_{a,p} = 1$ . If  $T_{a,p} = 1$ , then we denote  $N_{a,p}$  to be the smallest integer such that  $f_{\text{acc}}^{N_{a,p}} \geq 1 - \tau$ . An equivalent way of writing this condition is  $f(x^{N_{a,p}}) \leq f(x^*) + \tau(f(x^0) - f(x^*))$ . Successfully solving a problem with a tolerance of  $\tau = 0$  implies that the algorithm found the best solution.

The performance profile is a two-dimensional graph with one curve for each algorithm of the set  $\mathcal{A}$ . The horizontal axis corresponds to a ratio of number of function evaluations  $\alpha \geq 1$  and the vertical axis represents the corresponding proportion of problems solved within the tolerance  $\tau$ . More precisely, define

$$r_{a,p} = \begin{cases} \frac{N_{a,p}}{\min\{N_{\tilde{a},p} : \tilde{a} \in \mathcal{A}, T_{\tilde{a},p} = 1\}} & \text{if } T_{a,p} = 1, \\ \infty, & \text{if } T_{a,p} = 0. \end{cases}$$

Notice the algorithm  $a$  that successfully solves problem  $p$  with the least number of function evaluations will have  $r_{a,p} = 1$ . Since the number of function evaluations used is an integer, it is entirely possible for multiple algorithms to have  $r_{a,p} = 1$  on some problems. All other algorithms will have a value of  $r_{a,p}$  strictly greater than 1. An algorithm that successfully solved the problem using exactly twice as many function evaluations as the algorithm that solved it with the fewest evaluations will have  $r_{a,p} = 2$ . An algorithm that fails to solve the problem will have  $r_{a,p} = \infty$ .

With this notation, we can finally define the performance profile.

**DEFINITION A.1 (Performance Profile).**

*The portion of problems successfully solved by algorithm  $a \in \mathcal{A}$  on the test set  $\mathcal{P}$  is defined as the function  $\rho_a : [1, \infty) \mapsto [0, 1]$  with*

$$\rho_a(\alpha) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} : r_{a,p} \leq \alpha\}|,$$

*where  $|\cdot|$  counts the number of elements in a set. The performance profile is created by plotting the function  $\rho_a$  for all algorithms of  $\mathcal{A}$  on the same graph.*

Thus,  $\rho_a(\alpha)$  is the portion of problems that algorithm  $a$  solved within a ratio of  $\alpha$  function evaluations of the best solver for the respective problem.

**EXAMPLE A.1.** Consider the following fictional data.<sup>1</sup> Suppose that we ran 3 algorithms on 20 test problems with dimension ranging from 2 to 25. Each time a trial point improves the current best solution, its evaluation number  $N$  and objective function value  $f(x^N)$  are recorded. Table A.1 shows the dimension  $n_p$  of each problem as well as the number of function evaluations  $N_{a,p}$  required by algorithm  $a \in \mathcal{A}$  to solve problem  $p \in \mathcal{P}$  within tolerance  $\tau = 10\%$ . The table also reports the corresponding accuracy value  $f_{\text{acc}}^{N_{a,p}}$ .

<sup>1</sup>The reason we use fictional data is to better illustrate some of the strengths and weaknesses of profiles.

Coincidentally, each algorithm terminates after exactly  $N_{a,p}$  function evaluations (so  $f_{\text{acc}}^{N_{a,p}}$  is the highest accuracy ever obtained by the algorithm).<sup>2</sup> Note that Algorithm  $a = 1$  failed to solve Problem  $p = 6$  within accuracy  $\tau$ .

TABLE A.1. Fictional data generated by applying 3 algorithms on 20 test problems with  $\tau = 10\%$

| Problem number | Dim.<br>$n_p$ | Algorithm 1 |                            | Algorithm 2 |                            | Algorithm 3 |                            |
|----------------|---------------|-------------|----------------------------|-------------|----------------------------|-------------|----------------------------|
|                |               | $N_{a,p}$   | $f_{\text{acc}}^{N_{a,p}}$ | $N_{a,p}$   | $f_{\text{acc}}^{N_{a,p}}$ | $N_{a,p}$   | $f_{\text{acc}}^{N_{a,p}}$ |
| 1              | 2             | 15          | 0.959                      | 16          | 1.000                      | 22          | 0.993                      |
| 2              | 2             | 22          | 0.954                      | 25          | 0.996                      | 62          | 1.000                      |
| 3              | 2             | 37          | 0.969                      | 33          | 0.970                      | 25          | 1.000                      |
| 4              | 2             | 22          | 0.948                      | 15          | 1.000                      | 32          | 0.972                      |
| 5              | 3             | 48          | 0.944                      | 28          | 1.000                      | 35          | 0.956                      |
| 6              | 3             | -           | -                          | 84          | 1.000                      | 38          | 0.901                      |
| 7              | 3             | 22          | 0.940                      | 35          | 1.000                      | 55          | 0.969                      |
| 8              | 5             | 98          | 0.909                      | 47          | 1.000                      | 98          | 0.908                      |
| 9              | 5             | 43          | 0.967                      | 65          | 0.999                      | 29          | 1.000                      |
| 10             | 5             | 12          | 0.961                      | 91          | 1.000                      | 63          | 0.990                      |
| 11             | 7             | 28          | 0.996                      | 65          | 0.997                      | 61          | 1.000                      |
| 12             | 9             | 100         | 0.925                      | 56          | 1.000                      | 134         | 0.961                      |
| 13             | 9             | 72          | 0.927                      | 82          | 1.000                      | 53          | 0.944                      |
| 14             | 11            | 84          | 0.988                      | 200         | 0.997                      | 142         | 1.000                      |
| 15             | 12            | 33          | 0.982                      | 178         | 1.000                      | 165         | 0.995                      |
| 16             | 15            | 115         | 0.973                      | 255         | 0.996                      | 200         | 1.000                      |
| 17             | 16            | 19          | 0.983                      | 32          | 1.000                      | 28          | 0.980                      |
| 18             | 17            | 102         | 1.000                      | 205         | 0.996                      | 175         | 0.962                      |
| 19             | 20            | 305         | 0.909                      | 280         | 1.000                      | 391         | 0.913                      |
| 20             | 25            | 200         | 0.927                      | 111         | 1.000                      | 124         | 0.929                      |

The left part of Figure A.2 provides the performance profile for tolerance  $\tau = 10\%$  based on the data from Table A.1. The right part of the figure shows the performance profile for  $\tau = 5\%$ . The latter cannot be deduced from the tabular data presented here, as one would need to recompute the values  $N_{a,p}$  by inspecting the logs of the runs.

Let us analyze the left graph in Figure A.2, the performance profile with tolerance  $\tau = 10\%$ . For  $\alpha = 1$ ,  $\rho_a(1)$  gives the number of problems that algorithm  $a$  solved using the fewest number of evaluations (including ties). Algorithm 1 was the best of the three algorithms on 50% of the problems. In the left part of Figure A.2, Algorithms 2 and 3 solve every test problem, while Algorithm 1 failed to solve one problem to a tolerance of  $\tau = 10\%$ . The

<sup>2</sup>This is a highly unrealistic outcome, but valuable for the purpose of illustration and allows us to present a single table rather than one for  $\tau = 10\%$  and a second for  $\tau = 5\%$ .

figure shows that  $\rho_3(2) = 60\%$ , which means that 60% of the test problems were solved by Algorithm 3 within a tolerance of  $\tau = 10\%$  in less than or equal to twice the number of function calls used by the best of the three algorithms. For  $\alpha$  sufficiently large,  $\rho_a(\alpha)$  gives the portion of problems that algorithm  $a$  solved.

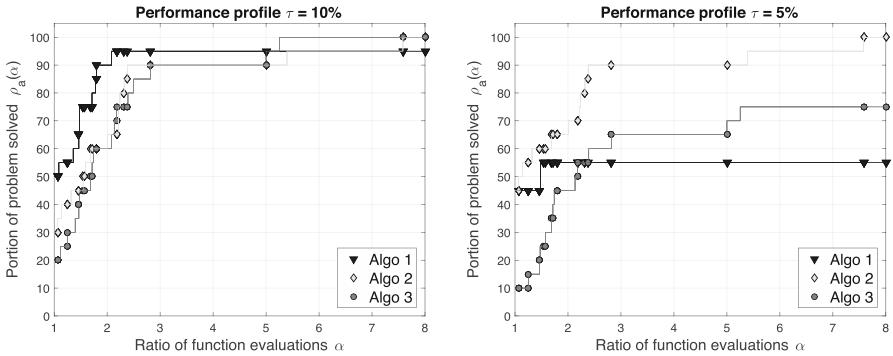


FIGURE A.2. Performance profiles for  $\tau = 10\%$  and  $\tau = 5\%$

In general, we seek algorithms whose performance function starts high, rises rapidly, and reaches end values close to 100%. Based on the profile in Figure A.2 with  $\tau = 10\%$ , we would either select Algorithm 1 (if speed was more important than robustness or if only a small budget of function evaluations were available), or Algorithm 3 (if robustness were more important than speed).

One of the weaknesses of performance profiles is the need to select a value of the tolerance parameter  $\tau$ . Sometimes changing this value can significantly alter the conclusions drawn from a performance profile. The right graph in Figure A.2 provides the performance profile using the tolerance  $\tau = 5\%$ . This smaller value makes it more difficult to pass the test of being successfully solved. With this stricter condition, Algorithm 2 stands out as being the best of the three algorithms. Its performance value dominates the others.

In conclusion, the performance with  $\tau = 10\%$  suggests that Algorithm 1 is the fastest to generate a solution within the vicinity of a minimiser, but the profile with  $\tau = 5\%$  suggests that Algorithm 2 is the best one to generate this higher accuracy solution.

---

On a final note, if the values of  $\alpha$  often get very large before each algorithm's performance reaches its maximum, then it can be useful to plot a performance profile using a log-scale on the  $x$ -axis. Our data is constructed in a manner that does not require this.

**A.3.3. Data Profiles.** *Data profiles* examine the efficiency and robustness of an algorithm from a different perspective than performance profiles. The information used to create data profiles is similar to the information for

performance profiles. Given the sets  $P$  and  $A$ , and a tolerance  $\tau$ , we construct Boolean variables  $T_{a,p}$  that indicate whether algorithm  $a$  successfully solved problem  $p$  within tolerance  $\tau$ . If  $T_{a,p} = 1$  (i.e., algorithm  $a$  solved problem  $p$  within tolerance  $\tau$ ), then we determine the number of function evaluations used up to the end of the iteration when the algorithm first solves the problem. This is denoted  $N_{a,p}$ . Finally, data profiles require the dimension of problem  $p \in P$ , denoted  $n_p$ .

**DEFINITION A.2 (Data Profile).**

*The portion of problems successfully solved by algorithm  $a \in \mathcal{A}$  on the test set  $\mathcal{P}$  within a budget  $k \in \mathbb{N}$  of groups of function evaluations is defined as the function  $d_a : \mathbb{N} \mapsto [0, 1]$  with*

$$d_a(k) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} : N_{a,p} \leq k(n_p + 1)T_{a,p}\}|$$

*where  $|\cdot|$  counts the number of elements in a set. The data profile is created by plotting the function  $d_a$  for all algorithms of  $\mathcal{A}$  on the same graph.*

Notice that  $d_a(k)$  is the proportion of problems solved within  $k$  groups of  $n_p+1$  function evaluations. The  $n_p+1$  term is inserted on the assumption that the number of function evaluations required is likely to increase as the number of variables increases, and  $n_p+1$  represents the number of function evaluations required in  $\mathbb{R}^{n_p}$  to construct a simplex gradient (see Definition 9.5).

**EXAMPLE A.2.** Figure A.3 provides data profiles with tolerance parameter  $\tau = 10\%$  and  $\tau = 5\%$ . Like Example A.1, the profiles are generated using Table A.1 under the assumptions that  $N_{a,p}$  is the number of function evaluations used to first solve the problem within a tolerance of  $\tau = 10\%$ , and that the algorithm terminates after exactly  $N_{a,p}$  function evaluations.<sup>3</sup>

The  $\tau = 10\%$  data profile reveals that  $d_1(10) = 70\%$ , i.e., 70% of the test problems were solved by Algorithm 1 within a tolerance of 10% in less than 10 groups of  $n_p + 1$  function evaluations.

Like performance profiles, algorithms whose graphs are higher are generally the more successful methods. The algorithms that rise rapidly are those that solve many problems quickly. Algorithms that end high are those that solve many problems successfully. In the case where  $\tau = 10\%$ , Algorithm 1 rises the fastest, but stops at 95%, while Algorithms 2 and 3 rise more slowly, but eventually solve all problems.

As with performance profiles, one weakness of data profiles is the need to select a definition for “successfully solved”, and changing this definition can

---

<sup>3</sup>This is still unrealistic, and still useful for illustration.

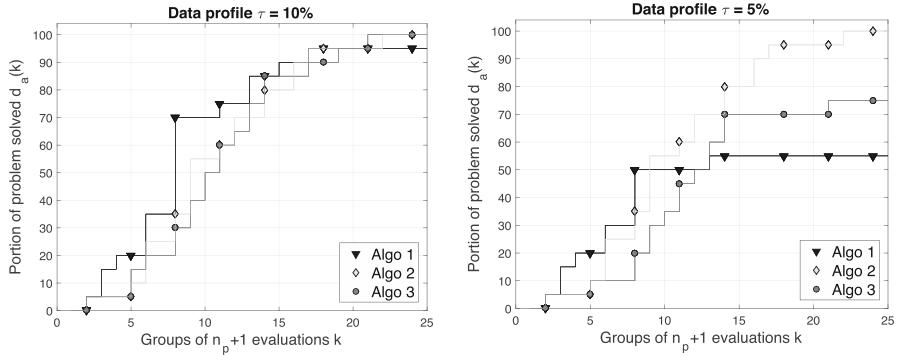


FIGURE A.3. Data profile based for  $\tau = 10\%$  and  $\tau = 5\%$

significantly alter the data profiles. For example, in Figure A.3 with  $\tau = 5\%$ , Algorithm 2 appears much stronger. As in Figure A.2, based on this data profile, we would either recommend Algorithm 1 if the number of available function evaluations was limited, or Algorithm 2 if robustness was the most important factor.

In the situation where the same best known value  $f(x^*)$  is used in all plots, the data profile for a given algorithm  $a$  is independent of the other algorithms. This means that, unlike performance profiles, if one algorithm is removed from the profile, then remaining profiles will not change.

Like performance profiles, it is sometimes useful to plot data profiles using a logarithmic scale on the  $x$ -axis.

**A.3.4. Accuracy Profiles.** Performance and data profiles both use two-dimensional plots to examine efficiency and robustness of algorithms. *Accuracy profiles* study robustness and quality of the final solution. We denote the total number of function evaluations used by algorithm  $a \in \mathcal{A}$  on problem  $p \in \mathcal{P}$  by  $N_{a,p}^{tot}$ . The final solution accuracy is  $f_{acc}^{N_{a,p}^{tot}}$  (see Equation (A.1)).

The negative of the base 10 logarithm of  $1 - f_{acc}^{N_{a,p}^{tot}}$  gives an indication of the number of correct decimals of the accuracy value. Indeed, if  $d \geq 0$ , then

$$-\log_{10} \left( 1 - f_{acc}^{N_{a,p}^{tot}} \right) \geq d \quad \Leftrightarrow \quad f_{acc}^{N_{a,p}^{tot}} \geq 1 - 10^{-d}.$$

Using this, we create the accuracy profiles.

**DEFINITION A.3 (Accuracy Profile).**

The portion of problems successfully solved by algorithm  $a \in \mathcal{A}$  to a relative accuracy of  $d$  on the test set  $\mathcal{P}$  is defined as the function  $r_a : [0, \infty) \mapsto [0, 1]$  with

$$r_a(d) = \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : -\log_{10} \left( 1 - f_{\text{acc}}^{N_{a,p}^{\text{tot}}} \right) \geq d \right\} \right|,$$

where  $-\log_{10}(0)$  is interpreted as  $\infty$ . The accuracy profile is created by plotting the function  $r_a$  for all algorithms of  $\mathcal{A}$  on the same graph.

Like the previous profiles, graphs that are consistently higher correspond to algorithms that consistently produce better results. However, in accuracy profiles, the graphs will begin at  $r_a(0) = 100\%$ , and will be nonincreasing (whereas in performance and data profiles the graphs are nondecreasing).

**EXAMPLE A.3.** Figure A.4 provides an example of an accuracy profile, based on the data in Table A.1, again with the (unrealistic) assumption that each algorithm terminated at exactly  $N_{a,p}$  function calls, so  $N_{a,p}^{\text{tot}} = N_{a,p}$ . In Table A.1 the entry  $f_{\text{acc}}^{N_{1,6}^{\text{tot}}}$  is missing, because the algorithm's accuracy never reached the threshold  $\tau = 10\%$ . The final accuracy value produced by Algorithm 1 on Problem 6 is  $f_{\text{acc}}^{N_{1,6}^{\text{tot}}} = 0.895$ , and required to plot the accuracy profile.

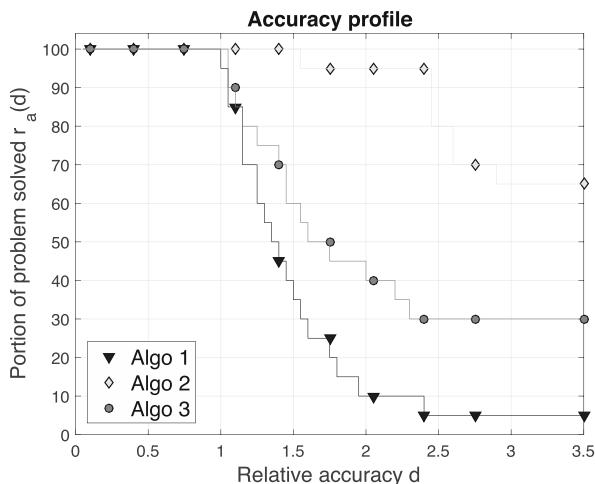


FIGURE A.4. Accuracy profile based on the data in Table A.1

The graph shows that  $r_1(2) = 10\%$ ; this can be interpreted as Algorithm 1 solves 10% of problems to a relative accuracy of 2 or more. In Figure A.4, Algorithm 2 appears to be the best, as it always solves the most problems for each level of accuracy.

---

Accuracy profiles address the weakness of having to select a definition for “successfully solved”. However, they introduce a new weakness. In particular, it is important to stress that accuracy profiles ignore the number of function evaluations required to achieve the presented results, and thus accuracy profiles can be strongly biased when different algorithms use significantly different numbers of function evaluations. This is the case for Figure A.4, where Algorithms 2 and 3 use higher numbers of function evaluations than Algorithm 1.

Accuracy profiles are much better suited for analyzing fixed-cost data sets, i.e., to compare algorithms where the overall budget of function evaluations is fixed:  $N_{a,p}^{tot}$  does not depend on  $a \in \mathcal{A}$ .

Another example of an accuracy profile can be found in Figure 13.4. In that figure, the  $x$ -axis is in logarithmic scale, and the profile compares three algorithms on a single optimization problem from 20 different starting points.

## EXERCISES

Exercises that require the use of a computer are tagged with a box symbol ( $\square$ ). More difficult exercises are marked by a plus symbol (+).

**EXERCISE A.1.** The values on the performance profiles in Figure A.2 when  $\alpha$  is large appear to be identical to the values of the data profiles in Figure A.3 when  $k$  is large. Is this always the case for any pair of performance and data profiles (using the same  $\tau$ )?

**EXERCISE A.2.** Examine the convergence plot in Figure A.1. Which algorithm is most likely to be a heuristic method (Part 2). Explain your answer.

**EXERCISE A.3.** Examine the trajectory plot in Figure A.1. Which algorithm is most likely to be a direct search method (Part 3). Explain your answer.

**EXERCISE A.4.**  $\square$  Use the data from Table A.1 to recreate the performance profile with tolerance  $\tau = 10\%$  of Figure A.2.

**EXERCISE A.5.**  $\square$  Use the data from Table A.1 to recreate the data profile with tolerance  $\tau = 10\%$  of Figure A.3.

**EXERCISE A.6.**  $\square$  Use the data from Table A.1 to recreate the accuracy profile of Figure A.4.

**EXERCISE A.7.** Provide an analysis of the performance profiles in Figure A.5.

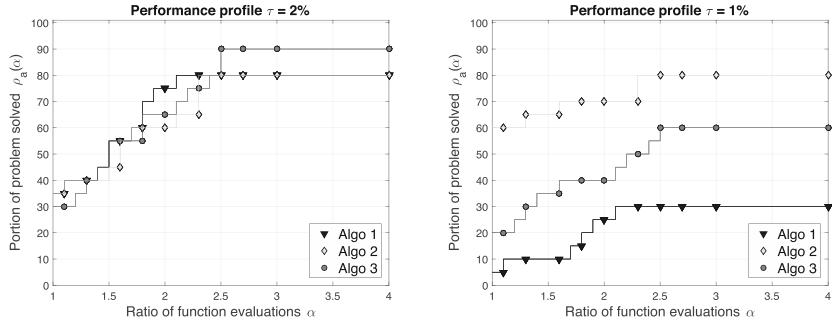


FIGURE A.5. Performance profile for Appendix exercises.

**EXERCISE A.8.** Provide an analysis of the data profiles in Figure A.6.

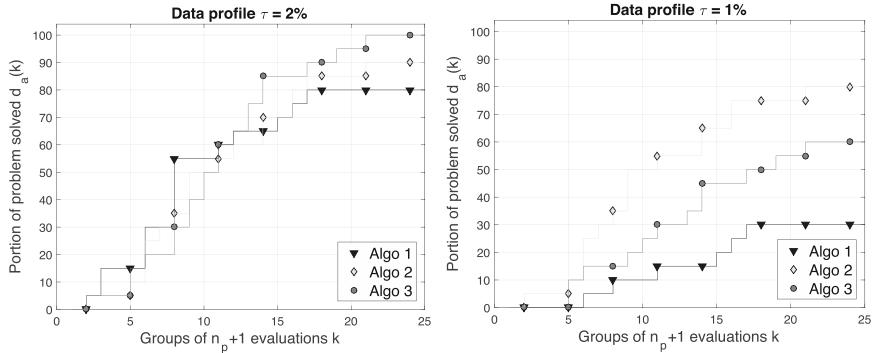


FIGURE A.6. Data profile for Appendix exercises.

**EXERCISE A.9.** Provide an analysis of the accuracy profile in Figure A.7.

**EXERCISE A.10.**  $\blacksquare$  Three fictional algorithms were ran on 20 test problems and produced the following logs. The best objective function values found by evaluation  $N$  with algorithm  $a \in \{1, 2, 3\}$  on problem  $p \in \{1, 2, \dots, 20\}$  appear in the following table.

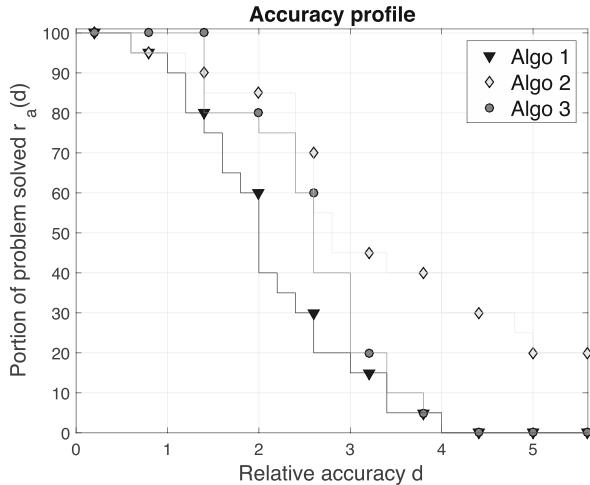


FIGURE A.7. Accuracy profile for Appendix exercises.

| Algorithm 1 |                   | Algorithm 2 |                           | Algorithm 3 |                        |
|-------------|-------------------|-------------|---------------------------|-------------|------------------------|
| $N$         | $f(x^N)$          | $N$         | $f(x^N)$                  | $N$         | $f(x^N)$               |
| 0           | $p$               | 0           | $p$                       | 0           | $p$                    |
| 1           | $\frac{p}{1+p}$   | 1           | $\frac{p}{2\sqrt{p}}$     | 10          | $\frac{p}{(1+p)^{10}}$ |
| $\vdots$    | $\vdots$          | $\vdots$    | $\vdots$                  | $\vdots$    | $\vdots$               |
| $2^i - 1$   | $\frac{p}{1+i p}$ | $i^3$       | $\frac{p}{(1+i)\sqrt{p}}$ | $10i^2$     | $\frac{p}{(1+p)^i}$    |
| $\vdots$    | $\vdots$          | $\vdots$    | $\vdots$                  | $\vdots$    | $\vdots$               |
| 1023        | $\frac{p}{1+10p}$ | 1000        | $\frac{p}{11\sqrt{p}}$    | 1000        | $\frac{p}{(1+p)^{10}}$ |

In the following questions, use logarithmic scales on the abscissa or ordinate when necessary.

- Trace the convergence plots for problem  $p = 1$  and for problem  $p = 10$ .
- Trace the performance profiles with  $\tau = 10^{-1}$  and  $\tau = 10^{-2}$ .
- Given that problem  $p$  contains  $n = 10 + p$  variables, trace the data profiles with  $\tau = 10^{-1}$  and  $\tau = 10^{-2}$ .
- Trace the accuracy profile.
- Make recommendations on which algorithm is preferable to use on these problems.



**Appendix Project: Benchmarking.** Implement one algorithm from each of Parts 1, 2, 3, and 4 of this book. Run each algorithm on the test problems below. Prepare a report that analyzes the data collected, making use of the tools described in this appendix. Include remarks on any weaknesses of the testing.

The problems below are the 7 test problems from Winfield's Ph.D. thesis [168] (which marks one of the first times a DFO method was systematically tested). Code for all of them can be found in the CUTEr/est Test Set [158]. Each problem takes the form

$$\min_x \{f(x) : x \in \Omega\}$$

starting from initial point  $x^0$  using initial sampling radius/mesh size  $\Delta^0$  (as applicable). The point  $x^*$  is the (approximate) location of the true minimiser of the problem.

(1) (Rosenbrock)

$$\begin{aligned} f(x) &= 100(x_2 - (x_1)^2)^2 + (1 - x_1)^2, \\ \Omega &= \mathbb{R}^2, x^0 = [-1.2, 1.0]^\top, \Delta^0 = 0.1, \\ x^* &= [1, 1]^\top \end{aligned}$$

(2) (Cube)

$$\begin{aligned} f(x) &= 100(x_2 - (x_1)^3)^2 + (1 - x_1)^2, \\ \Omega &= \mathbb{R}^2, x^0 = [-1.2, 1.0]^\top, \Delta^0 = 0.1, \\ x^* &= [1, 1]^\top \end{aligned}$$

(3) (Beale)

$$\begin{aligned} f(x) &= (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 (x_2)^2)^2 + (2.625 - x_1 + x_1 (x_2)^3)^2, \\ \Omega &= \mathbb{R}^2, x^0 = [0.1, 0.1]^\top, \Delta^0 = 0.5, \\ x^* &= [3, 0.5]^\top \end{aligned}$$

(4) (Modified Box)

$$\begin{aligned} f(x) &= \sum_{y \in Q} [x_3(e^{-x_1 y} - e^{-x_2 y}) - (e^{-y} - e^{-10y})]^2, \text{ where} \\ Q &= \{0.1, 0.2, \dots, 1.0\}, \\ \Omega &= [0, 25]^3, x^0 = [0, 10, 20]^\top, \Delta^0 = 0.5, \\ x^* &= [1, 10, 1]^\top \end{aligned}$$

(5) (Enzyme/Kowalik & Osborne I)

$$\begin{aligned} f(x) &= \sum_{i=1}^{11} \left( y_i - \frac{x_1(u_i^2 + u_i x_2)}{u_i^2 + u_i x_3 + x_4} \right)^2, \text{ where} \\ y_1 &= 0.1957 \quad u_1 = 4.0000 \quad y_7 = 0.0456 \quad u_7 = 0.1250 \\ y_2 &= 0.1947 \quad u_2 = 2.0000 \quad y_8 = 0.0342 \quad u_8 = 0.1000 \\ y_3 &= 0.1735 \quad u_3 = 1.0000 \quad y_9 = 0.0323 \quad u_9 = 0.0833 \\ y_4 &= 0.1600 \quad u_4 = 0.5000 \quad y_{10} = 0.0235 \quad u_{10} = 0.0714 \\ y_5 &= 0.0844 \quad u_5 = 0.2500 \quad y_{11} = 0.0246 \quad u_{11} = 0.0625 \\ y_6 &= 0.0627 \quad u_6 = 0.1670 \\ \Omega &= [0, 1]^4, x^0 = [0, 0, 0, 0]^\top, \Delta^0 = 0.1, \\ x^* &\approx [0.1927311, 0.1942269, 0.1244938, 0.1371652]^\top \end{aligned}$$

- (6) (Enzyme/Kowalik & Osborne II)

Same as the Enzyme/Kowalik & Osborne I, except

$$x^0 = [0.250, 0.390, 0.415, 0.390]^\top, \Delta^0 = 0.5,$$

$$x^* \approx [0.1927311, 0.1942269, 0.1244938, 0.1371652]^\top$$

- (7) (Powell Four Variable)

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4,$$

$$\Omega = \mathbb{R}^4, x^0 = [3, -1, 0, 1]^\top, \Delta^0 = 0.35,$$

$$x^* = [0, 0, 0, 0]^\top.$$

# *Solutions to Selected Exercises*

## **Solutions to Problems of Chapter 1**

- 1.2 a) Continuous, smooth.  
b) Discrete.  
c) Continuous, nonsmooth.  
d) Discrete.
- 1.4 a)  $\min_{[x,\alpha] \in \mathbb{R}^n \times \mathbb{R}} \{\alpha : \alpha \geq f(x), c(x) \leq 0\}$   
 $[\hat{x}, \hat{\alpha}] \in \operatorname{argmin}_{[x,\alpha] \in \mathbb{R}^n \times \mathbb{R}} \{\alpha : \alpha \geq f(x), c(x) \leq 0\}$   
 $\Leftrightarrow \hat{x} \in \operatorname{argmin}_{x \in \mathbb{R}^n} \{f(x) : c(x) \leq 0\}.$
- b)  $\beta_i \geq |x_i| \Leftrightarrow -\beta_i \leq x_i \leq \beta_i \Leftrightarrow (\beta_i \geq x_i \text{ and } \beta_i \geq -x_i).$   
c)  $\min_{(x,\beta) \in \mathbb{R}^{2n}} \{\sum_{i=1}^n \beta_i : \beta_i \geq x_i, \beta_i \geq -x_i \forall i \in \{1, \dots, n\}, c(x) \leq 0\}.$
- 1.6 a) Continuous, smooth.  
b) Continuous, nonsmooth.  
c) Discrete.

## **Solutions to Problems of Chapter 2**

- 2.1 a)  $\det(A) = -20.$   
b)  $\|A\| = \|A\|_2 = 5, \|A\|_F = \sqrt{43}.$   
c)

$$A^{-1} = \begin{bmatrix} -1/4 & 1/2 & 0 \\ 3/4 & -1/2 & 0 \\ 0 & 0 & 1/5 \end{bmatrix}.$$

- 2.2 a)  $\nabla f(\mathbf{0}) = [1, 2, 3, 4, 5]^\top$  and  $\nabla f(5, 4, 3, 2, 1) = (1, 2, 3, 4, 5)^\top \times e^{35}$ .  
 b)  $f'(x; v) = 35$ .  
 c)  $f'(x; v) = 0$ .
- 2.5  $\{d \in \mathbb{R}^3 : (e-2)d_1 + 5d_2 - 2d_3 < 0\}$ .
- 2.8 a) Closed, bounded, compact.  
 b) Open, closed.  
 c) Open, closed, bounded, compact.  
 d) Closed, bounded, compact.  
 e) None.
- 2.9 a)  $\text{cl}(\mathbb{R}^n) = \text{int}(\mathbb{R}^n) = \mathbb{R}^n$ .  
 b)  $\text{cl}(\emptyset) = \text{int}(\emptyset) = \emptyset$ .  
 c)  $\text{cl}(C) = \{x : 1 \leq \sum_{i=1}^n (x_i)^2 \leq 9\}$ ,  $\text{int}(C) = \{x : 1 < \sum_{i=1}^n (x_i)^2 < 9\}$ .  
 d)  $\text{cl}(D) = D$ ,  $\text{int}(D) = \emptyset$ .  
 e)  $\text{cl}(E) = \{x \in \mathbb{R}^3 : (x_1)^2 + (x_2)^2 \leq (x_3)^2, x_3 \geq 0\}$ ,  
 $\text{int}(E) = \{x \in \mathbb{R}^3 : (x_1)^2 + (x_2)^2 < (x_3)^2, x_3 > 0\}$ .
- 2.14 a)  $\text{vol}(\mathbb{Y}) = \frac{1}{6}$ ,  $\text{von}(\mathbb{Y}) = \frac{\sqrt{2}}{24}$ .  
 b)  $\text{vol}(\mathbb{Y}) = \frac{32}{3}$ ,  $\text{von}(\mathbb{Y}) = \frac{\sqrt{2}}{24}$ .  
 c)  $\text{vol}(\mathbb{Y}) = \frac{4}{3}$ ,  $\text{von}(\mathbb{Y}) = \frac{4}{519\sqrt{173}}$  almost degenerate.

### Solutions to Problems of Chapter 3

- 3.4 a)  $p \geq \frac{10^3}{2} + 1$ .  
 b)  $p^2 \geq (\sqrt{2}\frac{10^3}{2} + 1)^2$ .  
 c)  $p^n \geq (\sqrt{n}\frac{10^3}{2} + 1)^n$ .  
 d)  $n \leq 6$ .
- 3.7 Complete:  $x^k = [0, -k]^\top$  with  $f(x^k) = -k \rightarrow -\infty$ .  
 Opportunist:  $x^k = [k, 0]^\top$  with  $f(x^k) = e^{-k} \rightarrow 0$ .  
 Opportunist with dynamic ordering: identical behaviour as the opportunistic strategy.  
 In all cases, the iterates diverge. This is because the level sets of  $f$  are unbounded.
- 3.11 a) Diamonds centred at the origin.  
 b) Let  $i \in \{1, 2\}$  be such that  $x_i^0 \neq 0$  and choose any  $\delta \in (0, |x_i^0|)$ .  
 Let  $j \neq i$  be the other index in  $\{1, 2\}$ .  
 If  $x_i^0 > 0$ , then  $f(x^0 - \delta e_i) = |x_i^0 - \delta| + |x_j^0| < |x_i^0| + |x_j^0| = f(x^0)$ .  
 If  $x_i^0 < 0$ , then  $f(x^0 + \delta e_i) = |x_i^0 + \delta| + |x_j^0| < |x_i^0| + |x_j^0| = f(x^0)$ .  
 c) The level sets of  $f$  are bounded, and  $f \in \mathcal{C}^0$ . By b), the only point where  $f'(x; \pm e_i)$ ,  $i = 1, 2$  are nonnegative is the origin  $\hat{x} = [0, 0]^\top$ . Theorem 3.4 ensures that the sequence of iterates produced by CS converges to  $\hat{x}$ , the unique global solution.

### Solutions to Problems of Chapter 4

4.2 a)  $[0, -1, 0] \rightarrow [0, 0, 0]$        $[2, -1, 0] \rightarrow [1, 0, 0]$   
 $[0, -1, 4] \rightarrow [0, 0, 1]$        $[2, 1, 0] \rightarrow [1, 1, 0]$   
 $[0, 1, 0] \rightarrow [0, 1, 0]$        $[2, -1, 4] \rightarrow [1, 0, 1]$   
 $[0, 1, 4] \rightarrow [0, 1, 1]$        $[2, 1, 4] \rightarrow [1, 1, 1]$ .

b)  $[0, -5/8, 15/4]$ .

4.4 a)  $P_{N=1}(x^2) = 1/5$ ,  $P_{N=2}(x^2) = 0.13$ ,  $P_{N=5}(x^2) = 1/10$ .  
b)  $P_{N=1}(x^2) = 1/5$ ,  $P_{N=2}(x^2) = 1/10$ ,  $P_{N=5}(x^2) = 0$ .

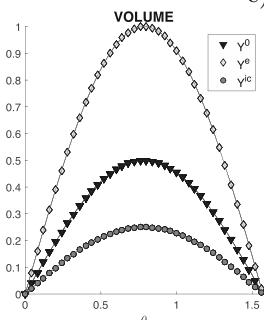
4.6 a)  $p = 0.03456$ .  
b)  $p = 0.9750$ .

### Solutions to Problems of Chapter 5

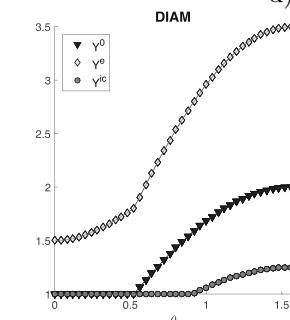
5.6 a) Reflect:  $\mathbb{Y}^{k+1} = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1/3 \\ 0 & 1 & 0 & -1/3 \end{bmatrix}$ ,  
shrink:  $\mathbb{Y}^{k+1} = \begin{bmatrix} 0 & 0 & 0 & \gamma \\ 0 & 0 & \gamma & \gamma \\ 0 & \gamma & 0 & \gamma \end{bmatrix}$ ,  
others:  $\mathbb{Y}^{k+1} = \begin{bmatrix} 0 & 0 & 0 & -\delta \\ 0 & 0 & 1 & (1-2\delta)/3 \\ 0 & 1 & 0 & (1-2\delta)/3 \end{bmatrix}$   
with  $\delta \in \{\delta^e, \delta^{ic}, \delta^{oc}\}$ .  
b)  $\text{vol}(\mathbb{Y}^k) = \frac{1}{6}$ ,  $\text{vol}(\mathbb{Y}^{k+1}) = \frac{1}{6}$ ,  $\text{von}(\mathbb{Y}^k) = \frac{1}{6\sqrt{3}^3}$ ,  $\text{von}(\mathbb{Y}^{k+1}) = \frac{1}{6\sqrt{\frac{26}{9}}^3}$ .

5.9 a)  $|\text{vol}(\mathbb{Y}^r)| = |\delta^r| \text{vol}(\mathbb{Y}^0) = \text{vol}(\mathbb{Y}^0)|$ ,  
 $\text{diam}(\mathbb{Y}^0) = \text{diam}(\mathbb{Y}^r) = \max\{1, 2|\sin \theta|\}$ .  
b)  $|\text{vol}(\mathbb{Y}^{ic})| = |\delta^{ic}| \text{vol}(\mathbb{Y}^0) = |\delta^{oc}| \text{vol}(\mathbb{Y}^0) = \text{vol}(\mathbb{Y}^{ic})|$ ,  
 $\text{diam}(\mathbb{Y}^{ic}) = \text{diam}(\mathbb{Y}^{oc}) = \max\{\sqrt{\frac{1}{4} \cos^2(\theta) + \sin^2(\theta)}, 2|\sin \theta|\}$ .

c)



d)



e)



5.12 a)  $\text{vol}(\mathbb{Y}^k) = \frac{1}{2} |\lambda^{k+1} \mu^k - \lambda^k \mu^{k+1}| = \sqrt{33}/2^{k+3}$   
 $\text{vol}(\mathbb{Y}^{k+1})/\text{vol}(\mathbb{Y}^k) = 1/2$ .

- b)  $\text{diam}(\mathbb{Y}^k) = \sqrt{\lambda^{2k} + \mu^{2k}} \rightarrow 0.$   
c)  $\text{von}(\mathbb{Y}^k) = \frac{\sqrt{33}}{2^{k+3}(\lambda^{2k} + \mu^{2k})} \rightarrow 0.$

### Solutions to Problems of Chapter 6

- 6.3 a)  $\{e_1, e_2, e_3, -e_1, -e_2, -e_3\}.$   
b)  $\{e_1, e_2, e_3, -\sum_{i=1}^2 e_i, -e_3\}.$   
c)  $\{e_1, e_2, e_3, -\sum_{i=1}^3 e_i\}.$   
d) Take the tetrahedron centred at  $(0, 0, 0)$ :  
 $\{[\sqrt{3}, 1, -1]^\top, [-\sqrt{3}, 1, -1]^\top, [0, -2, -1]^\top, [0, 0, 1]^\top\}.$
- 6.8 Let  $\tau = \theta\epsilon$  where  $\theta \in (0, 1]$

$$\begin{aligned}\frac{f(y + \tau d) - f(y)}{\tau} &= \frac{f(y + \theta\epsilon d) - f(y)}{\theta\epsilon} \\ &= \frac{f(\theta(y + \epsilon d) + (1 - \theta)y) - f(y)}{\theta\epsilon} \\ &\leq \frac{\theta f(y + \epsilon d) + (1 - \theta)f(y) - f(y)}{\theta\epsilon} = \frac{f(y + \epsilon d) - f(y)}{\epsilon}.\end{aligned}$$

- 6.13 a)

$$f(x) = \begin{cases} x_1 + x_2 & \text{if } x_1 \leq 0 \text{ and } x_2 \leq 0 \\ -2x_1 + x_2 & \text{if } 0 < x_1 \geq x_2 \\ x_1 - 2x_2 & \text{if } 0 < x_2 > x_1. \end{cases}$$

- b)  $f^\circ$  is always convex with respect to  $d$ .

### Solutions to Problems of Chapter 7

- 7.1 a)  $G = I$  and  $Z = D$ .  
b) Eight minimal bases:  
 $\pm\{e_1, -e_1 + e_2, -e_1 - e_2\}$  and  $\pm\{e_2, -e_2 + e_1, -e_2 - e_1\}$  and  
 $\pm\{e_1, e_2, -e_1 - e_2\}$  and  $\pm\{e_1, -e_2, -e_1 + e_2\}$ .  
Two equi-angle maximal positive bases  
 $\{e_1, e_2, -e_1, -e_2\}$  and  $\{(e_1 + e_2), (e_1 - e_2), -(e_1 + e_2), -(e_1 - e_2)\}$ .  
Four maximal positive bases  
 $\{e_1, -e_1, (e_1 + e_2), -(e_1 + e_2)\}$  and  $\{e_2, -e_2, (e_1 + e_2), -(e_1 + e_2)\}$  and  
 $\{e_1, -e_1, (e_1 - e_2), -(e_1 - e_2)\}$  and  $\{e_2, -e_2, (e_1 - e_2), -(e_1 - e_2)\}$ .
- 7.2 a)  $x^* = 10,000.01$ .  
b) 10,001.  
c) 53.  
7.4 a)  $\mathbb{D}_k$  contains the positive and negative coordinate vectors.  
b) Each poll step requires the evaluation of  $3^n - 1$  function evaluations.

### Solutions to Problems of Chapter 8

- 8.2 a)  $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\} = 1$  implies  $\delta^k \|d\|_\infty \leq \Delta^k b \Rightarrow \|d\|_\infty \leq \frac{1}{2^p} 4^p = 2^p$ . Thus  $d_1, d_2 \in P = \{-(2^p), -(2^p) + 1, \dots, 0, \dots, 2^p\}$ .  
 $|P| = 2(2^p) + 1 = 2^{p+1} + 1 \Rightarrow |\{d, d_1 \text{ and } d_2 \in P\}| = (2^{p+1} + 1)^2$ .  
 Removing  $[0, 0]^\top$  yields  $(2^{p+1} + 1)^2 - 1$  possibilities.  
 b)  $(2^{p+1} + 1)^n - 1$ .

- 8.5 Let  $\Omega = [0, 1]^2$  and  $f(x) = x_1 + x_2$ , so  $\operatorname{argmin}\{f(x, y), (x, y) \in \Omega\} = \{[0, 0]^\top\}$ . At  $[0, 0]^\top$ , the refined directions belong to  $\mathbb{R}_+^2 \setminus \{[0, 0]^\top\}$ .

8.9 a)  $H = \frac{1}{25} \begin{bmatrix} -7 & -24 \\ -24 & 7 \end{bmatrix}$ .  
 b)  $H = \frac{1}{169} \begin{bmatrix} 151 & -24 & 72 \\ -24 & 137 & 96 \\ 72 & 96 & -119 \end{bmatrix}$ .  
 c)  $H = \frac{1}{13} \begin{bmatrix} 9 & -4 & 6 & 6 \\ -4 & 9 & 6 & 6 \\ 6 & 6 & 4 & -9 \\ 6 & 6 & -9 & 4 \end{bmatrix}$ .

### Solutions to Problems of Chapter 9

- 9.5 a) Yes, with constants  $|\lambda|\kappa_f$  and  $|\lambda|\kappa_g$ .  
 b) No. Consider for example  $f(x) = x$  and  $\tilde{f}_\Delta(x) = x + \Delta^2$ .  
 9.7 a) The order of the vectors only affects the orders of the rows in the system  $[\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y})$ .  
 b) If  $\mathbb{Y}$  is poised for linear regression, then the rank of  $[\mathbf{1} \ Y^\top] \begin{bmatrix} \alpha_0 \\ \alpha \end{bmatrix} = f(\mathbb{Y})$  is  $n + 1$ . Thus the matrix is invertible.  
 Notice that  $\min_{\alpha_0, \alpha} \sum_{i=0}^n |\alpha_0 + \alpha^\top y^i - f(y^i)|^2 \geq 0$ , so the minimum value is reached when  $\alpha_0 + \alpha^\top y^i = f(y^i)$ ,  $\forall i = 0, 1, \dots, n$ .  
 9.8 a)  $\alpha \approx 0.9625$ .  
 b)  $\alpha = 1$ .  
 c)  $\alpha = 0.8125$   
 d) The true gradient is  $f'(\bar{x}) = 0.75$ .

### Solutions to Problems of Chapter 10

- 10.1 a)  $\tilde{g}_\Delta(x) = \nabla \tilde{f}_\Delta(x) = (x + \Delta + 1)$  and  $\nabla f(x) = e^x$ .  
 $\|\nabla f(0) - \tilde{g}_\Delta(0)\| = \|1 - (\Delta + 1)\| = \Delta$ .  
 b) Let  $y \in B_\Delta(0)$ , then

$$\lim_{\Delta \rightarrow 0} |e^y - \frac{1}{2}(y + \Delta + 1)^2 + 41| = |\frac{1}{2} + 41| > 0.$$

c) Because  $\tilde{f}_\Delta(0) \neq f(0)$ .

d) Let  $\tilde{f}'_\Delta(x) = \frac{1}{2}(x + \Delta + 1)^2 + K$ . Then  $\tilde{f}'_\Delta(0) = f(0) \Leftrightarrow K = \Delta(\frac{\Delta}{2} + 2)$ .

10.5 Let  $f(x) = x$  and  $d = -1$ . Then  $f(x + td) = x - t$  and  $f(x) + \eta t d \nabla f(x) = x - \eta t$ .

$$x - t < x - \eta t \Leftrightarrow \eta \in (0, 1).$$

10.10 a)  $0 < t < \frac{1}{2}$ .

b)  $0 < t \leq \frac{2}{2+\Delta} - \frac{1}{2}$ .

c)  $0 < t \leq \frac{2}{2-\Delta} - \frac{1}{2}$ .

### Solutions to Problems of Chapter 11

11.5  $x \approx [0.8052, -0.5931]^\top$ .

11.6

$$\begin{aligned}\tilde{f}_\Delta(x_c) &= \min_t \{\tilde{f}_\Delta(\hat{x} + t\tilde{g}) : \hat{x} + t\tilde{g} \in B_\Delta(\hat{x})\} \\ &= \min_t \{\phi(t) : \|\hat{x} - (\hat{x} + t\tilde{g})\| < \Delta\} \\ &= \min_t \{\phi(t) : \|t\tilde{g}\| < \Delta\} = \min_t \left\{ \phi(t) : t < \frac{\Delta}{\|\tilde{g}\|} \right\}.\end{aligned}$$

11.8 a)  $x_c = [\frac{1}{2}, 0]^\top$ ,  $f(x_c) = -\frac{1}{2}$ .  
b)  $-\frac{1}{2\sqrt{2}}$ .

### Solutions to Problems of Chapter 12

12.1  $Q(x) = 8.8 - 2.8(x - 59) + 1.7(x - 59)(x - 60)$ . The proposed point is  $\frac{11}{34}$ .

12.3 a) Define  $g(x) = \max(c_j(x), 0)^2$  and consider  $x \in \mathbb{R}^n$ :

$$\nabla g(x) = \begin{cases} 0 & \text{if } c_j(x) < 0 \\ 2c_j(x)\nabla c_j(x) & \text{if } c_j(x) > 0. \end{cases}$$

Both values of  $\nabla g \in \mathbb{R}^n$  converge to 0 as  $x \rightarrow 0$ . Therefore  $g \in \mathcal{C}^1$ .

b) Consider  $c(x) = x$ . Then  $h_1(x) = \max(x, 0)$  and  $\nabla h_1$  is not continuous at  $x = 0$ .

12.5 a)  $x^0 = [-1, -1]^\top$  with  $h(x^0) = 2$ ,  $f(x^0) = -2$ ,  $x^1 = [0, -1]^\top$  with  $h(x^1) = 1$ ,  $f(x^1) = -1$ ,  $x^2 = [0, 0]^\top$  with  $h(x^2) = 0$ ,  $f(x^2) = 0$ .

b)  $x^0 = [0, 0]^\top$  with  $h(x^0) = \infty$ ,  $f(x^0) = 0$  (notice the division by 0),  $x^k = [k, 0]^\top$  with  $h(x^k) = \frac{1}{k^2}$ ,  $f(x^k) = k$ . When  $k > \sqrt{10^{15}}$ , due to the machine precision  $x^k = [k, 0]^\top$  with  $h(x^k) = 0$ ,  $f(x^k) = k$  is feasible.

### Solutions to Problems of Chapter 13

13.1 There is no gain, because computational cost of both functions are identical.

13.3 a) Function  $f$ :

```

    INPUT  $x \in \mathbb{R}^7 \times \{0, 1\}^3$ 
 $t_0 = \text{TIME}()$ 
For  $i = 1$  to 20
    Launch the program to perform task  $i$ 
 $t_f = \text{TIME}()$ 
RETURN  $t_f - t_0$ .

```

- b)  $\Omega := \{x \in \mathbb{R}^7 \times \{0, 1\}^3 : 0 \leq x_i \leq 100, i \in \{1, 2, \dots, 7\}, x_8 + x_9 + x_{10} \leq 2\}$ .
- c)  $\min_x \{f(x) : x \in \Omega\}$ .

13.7 a) Compute the value  $f(x)$ . Launch the simulation if  $f(x) < f(x^k)$  otherwise redefine  $c(x) = \infty$ .

b) Yes

c) Use past values of  $c(x)$  to build a model  $\tilde{c}$  of  $c$ . Given a list of trial point  $\mathcal{L}$ , if  $\hat{x} \in \mathcal{L}$  has  $\tilde{c}(\hat{x}) > 0$ , then evaluate  $\hat{x}$  last.

### Solutions to Problems of Chapter 14

- 14.1 a) FALSE.
- b) FALSE.
- c) TRUE.
- d) FALSE.

14.5 Consider  $u, v \in \Omega$  with  $F(u) \neq F(v)$ .

$(\Rightarrow)$  If  $u \prec v$ , then  $f^{(1)}(u) \leq f^{(1)}(v)$  and  $f^{(2)}(u) \leq f^{(2)}(v)$  and therefore  $\phi_r(F(u)) = - (f^{(1)}(v) - f^{(1)}(u))^2 (f^{(2)}(v) - f^{(2)}(u))^2 \leq 0$ .

$(\Leftarrow)$  If  $\phi_r(F(u)) \leq 0$  where  $r = F(v)$ , then there are two possibilities.

First, if  $F(u) \leq F(v)$ , then  $u \prec v$  because  $F(u) \neq F(v)$ .

Second, if  $F(u) \not\leq F(v)$ , then

$$\begin{aligned}\phi_r(F(u)) &= \left( \max\{f^{(1)}(u) - f^{(1)}(v), 0\} \right)^2 + \left( \max\{f^{(2)}(u) - f^{(2)}(v), 0\} \right)^2 \\ &> 0\end{aligned}$$

which is a contradiction.

- 14.8 a)  $\Omega_{\mathcal{P}} = \{x \in \mathbb{R}^2 : x_1 + x_2 = 1, x \geq 0\}$ .
- b)  $F_{\mathcal{P}} = \{[3z, 2\sqrt{1-z}]^\top \in \mathbb{R}^2 : 0 \leq z \leq 1\}$ .
- c)  $x^1 = [0, 4]^\top$  and  $x^2 = [4, 0]^\top$  :  $F(x^1) = [0, 4]^\top$  and  $F(x^2) = [12, 0]^\top$ .

**Solutions to Problems of the Appendix**

- A.1 Yes, because it represents the portion of problem solved within  $\tau$ , regardless of the number of function evaluations.
- A.2 Algo 2 has few local improvements in the objective function value. This suggests far reaching sampling in the space of variables.
- A.3 The staircase behaviour of Algo 3 suggests usage of the CS algorithm.

# References

1. M.A. Abramson, Mixed variable optimization of a Load-Bearing thermal insulation system using a filter pattern search algorithm. *Optim. Eng.* **5**(2), 157–177 (2004)
2. M.A. Abramson, Second-order behavior of pattern search. *SIAM J. Optim.* **16**(2), 315–330 (2005)
3. M.A. Abramson, C. Audet, Convergence of mesh adaptive direct search to second-order stationary points. *SIAM J. Optim.* **17**(2), 606–619 (2006)
4. M.A. Abramson, C. Audet, J.W. Chrissis, J.G. Walston, Mesh adaptive direct search algorithms for mixed variable optimization. *Optim. Lett.* **3**(1), 35–47 (2009)
5. M.A. Abramson, C. Audet, G. Couture, J.E. Dennis Jr., S. Le Digabel, C. Tribes, The NOMAD project. Software available at <https://www.gerad.ca/nomad>
6. M.A. Abramson, C. Audet, J.E. Dennis Jr., S. Le Digabel, OrthoMADS: a deterministic MADS instance with orthogonal directions. *SIAM J. Optim.* **20**(2), 948–966 (2009)
7. H.L. Anderson, W.C. Davidon, M.G. Glicksman, U.E. Kruse, Scattering of positive pions by hydrogen at 189 MeV. *Phys. Rev.* **100**, 279–287 (1955)
8. C. Audet, Convergence results for generalized pattern search algorithms are tight. *Optim. Eng.* **5**(2), 101–122 (2004)
9. C. Audet, A short proof on the cardinality of maximal positive bases. *Optim. Lett.* **5**(1), 191–194 (2011)
10. C. Audet, A survey on direct search methods for blackbox optimization and their applications, in *Mathematics Without Boundaries: Surveys in Interdisciplinary Research*, ed. by P.M. Pardalos, T.M. Rassias, Chap. 2, pp. 31–56 (Springer, Berlin, 2014)
11. C. Audet, Tuning Runge-Kutta parameters on a family of ordinary differential equations. *Int. J. Math. Model. Numer. Optim.* (in press 2018)
12. C. Audet, J.E. Dennis Jr., Pattern search algorithms for mixed variable programming. *SIAM J. Optim.* **11**(3), 573–594 (2001)
13. C. Audet, J.E. Dennis Jr., Analysis of generalized pattern searches. *SIAM J. Optim.* **13**(3), 889–903 (2003)

14. C. Audet, J.E. Dennis Jr., A pattern search filter method for nonlinear programming without derivatives. *SIAM J. Optim.* **14**(4), 980–1010 (2004)
15. C. Audet, J.E. Dennis Jr., Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.* **17**(1), 188–217 (2006)
16. C. Audet, J.E. Dennis Jr., Nonlinear programming by mesh adaptive direct searches. *SIAG/Optim. Views-and-News* **17**(1), 2–11 (2006)
17. C. Audet, J.E. Dennis Jr., A progressive barrier for derivative-free nonlinear programming. *SIAM J. Optim.* **20**(1), 445–472 (2009)
18. C. Audet, V. Béchard, J. Chaouki, Spent potliner treatment process optimization using a MADS algorithm. *Optim. Eng.* **9**(2), 143–160 (2008)
19. C. Audet, G. Savard, W. Zghal, Multiobjective optimization through a series of single-objective formulations. *SIAM J. Optim.* **19**(1), 188–210 (2008)
20. C. Audet, S. Le Digabel, C. Tribes, NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD (2009)
21. C. Audet, G. Savard, W. Zghal, A mesh adaptive direct search algorithm for multiobjective optimization. *Eur. J. Oper. Res.* **204**(3), 545–556 (2010)
22. C. Audet, J.E. Dennis Jr., S. Le Digabel, Globalization strategies for mesh adaptive direct search. *Comput. Optim. Appl.* **46**(2), 193–215 (2010)
23. C. Audet, C.-K. Dang, D. Orban, Optimization of algorithms with OPAL. *Math. Program. Comput.* **6**(3), 233–254 (2014)
24. C. Audet, S. Le Digabel, M. Peyrega, Linear equalities in blackbox optimization. *Comput. Optim. Appl.* **61**(1), 1–23 (2015)
25. P. Balaprakash, S.M. Wild, B. Norris, Spapt: search problems in automatic performance tuning. *Procedia Comput. Sci.* **9**, 1959–1968 (2012). Proceedings of the International Conference on Computational Science, ICCS 2012
26. T. Begin, B. Baynat, F. Sourd, A. Brandwajn, A DFO technique to calibrate queueing models. *Comput. Oper. Res.* **37**(2), 273–281 (2010)
27. V. Beiranvand, W. Hare, Y. Lucet, Benchmarking of single-objective optimization algorithms. *Eng. Optim.* (to appear)
28. H.-G. Beyer, H.-P. Schwefel, Evolution strategies – a comprehensive introduction. *Nat. Comput.* **1**(1), 3–52 (2002)
29. K. Bigdeli, W. Hare, S. Tesfamariam, Configuration optimization of dampers for adjacent buildings under seismic excitations. *Eng. Optim.* **44**(12), 1491–1509 (2012)
30. S.C. Billups, J. Larson, P. Graf, Derivative-free optimization of expensive functions with computational error using weighted regression. *SIAM J. Optim.* **23**(1), 27–53 (2013)
31. M. Björkman, K. Holmström, Global optimization of costly nonconvex functions using radial basis functions. *Optim. Eng.* **1**, 373–397 (2000)
32. A.J. Booker, Well-conditioned Kriging models for optimization of computer simulations. Technical Report M&CT-TECH-00-002, Boeing Computer Services, Research and Technology, M/S 7L-68, Seattle, Washington 98124 (2000)
33. A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.W. Moore, D.B. Serafini, Managing surrogate objectives to optimize a helicopter rotor design – further experiments, in *AIAA Paper 1998-4717*, Presented at the *8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis (1998)
34. A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.B. Serafini, V. Torczon, Optimization using surrogate objectives on a helicopter test example, in *Optimal Design and Control*, ed. by J. Borggaard, J. Burns, E. Cliff, S. Schreck. *Progress in Systems and Control Theory* (Birkhäuser, Cambridge, MA, 1998), pp. 49–58

35. A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.B. Serafini, V. Torczon, M.W. Trosset, A rigorous framework for optimization of expensive functions by surrogates. *Struct. Multidiscip. Optim.* **17**(1), 1–13 (1999)
36. G.E.P. Box, Evolutionary operation: a method for increasing industrial productivity. *Appl. Stat.* **6**(2), 81–101 (1957)
37. Á. Bürmen, J. Puhan, T. Tuma, Grid restrained Nelder-Mead algorithm. *Comput. Optim. Appl.* **34**(3), 359–375 (2006)
38. P.J. Carreau, D. De Kee, R.P. Chhabra, *Rheology of Polymeric Systems* (PWS Kent, Boston, 1993)
39. X. Chen, C.T. Kelley, Optimization with hidden constraints and embedded Monte Carlo computations. *Optim. Eng.* **17**(1), 157–175 (2016)
40. T.D. Choi, O.J. Eslinger, C.T. Kelley, J.W. David, M. Etheridge, Optimization of automotive valve train components with implicit filtering. *Optim. Eng.* **1**(1), 9–27 (2000)
41. F.H. Clarke, *Optimization and Nonsmooth Analysis* (Wiley, New York, 1983). Reissued in 1990 by SIAM Publications, Philadelphia, as vol. 5 in the series Classics in Applied Mathematics
42. A.R. Conn, S. Le Digabel, Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optim. Methods Softw.* **28**(1), 139–158 (2013)
43. A.R. Conn, P.L. Toint, Nonlinear optimization and applications, in *An Algorithm using Quadratic Interpolation for Unconstrained Derivative Free Optimization* (Springer, Berlin, 1996), pp. 27–47
44. A.R. Conn, N.I.M. Gould, P.L. Toint, *Trust-Region Methods*. MPS-SIAM Series on Optimization (SIAM, Providence, 2000)
45. A.R. Conn, K. Scheinberg, L.N. Vicente, Geometry of interpolation sets in derivative free optimization. *Math. Program.* **111**(1–2), 141–172 (2008)
46. A.R. Conn, K. Scheinberg, L.N. Vicente, Geometry of sample sets in derivative free optimization: polynomial regression and underdetermined interpolation. *IMA J. Numer. Anal.* **28**(4), 721–749 (2008)
47. A.R. Conn, K. Scheinberg, L.N. Vicente, *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization (SIAM, Philadelphia, 2009)
48. I.D. Coope, C.J. Price, Frame-based methods for unconstrained optimization. *J. Optim. Theory Appl.* **107**(2), 261–274 (2000)
49. I.D. Coope, C.J. Price, Positive bases in numerical optimization. *Comput. Optim. Appl.* **21**(2), 169–175 (2002)
50. E.J. Cramer, J.M. Gablonsky, Effective parallel optimization of complex computer simulations, in *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (August 2004)
51. A.L. Custódio, J.F.A. Madeira, Glods: global and local optimization using direct search. *J. Glob. Optim.* **62**(1), 1–28 (2015)
52. A.L. Custódio, H. Rocha, L.N. Vicente, Incorporating minimum Frobenius norm models in direct search. *Comput. Optim. Appl.* **46**(2), 265–278 (2010)
53. A.L. Custódio, J.F.A. Madeira, A.I.F. Vaz, L.N. Vicente, Direct multisearch for multiobjective optimization. *SIAM J. Optim.* **21**(3), 1109–1140 (2011)
54. A.L. Custódio, K. Scheinberg, L.N. Vicente, Methodologies and software for derivative-free optimization, in *Advances and Trends in Optimization with Engineering Applications*, ed. by T. Terlaky, M.F. Anjos, S. Ahmed. MOS-SIAM Book Series on Optimization, Chap. 37 (SIAM, Philadelphia, 2017)

55. G.B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, 1963)
56. I. Das, J.E. Dennis Jr., Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.* **8**(3), 631–657 (1998)
57. C. Davis, Theory of positive linear dependence. *Am. J. Math.* **76**, 733–746 (1954)
58. J.E. Dennis Jr., V. Torczon, Direct search methods on parallel machines. *SIAM J. Optim.* **1**(4), 448–474 (1991)
59. J.E. Dennis Jr., V. Torczon, Managing approximation models in optimization, in *Multidisciplinary Design Optimization: State of the Art*, ed. by N.M. Alexandrov, M.Y. Hussaini (SIAM, Philadelphia, 1997), pp. 330–347
60. J.E. Dennis Jr., D.J. Woods, Optimization on microcomputers: the Nelder–Mead simplex algorithm, *New Computing Environments: Microcomputers in Large-Scale Computing*, ed. by A. Wouk (Society for Industrial and Applied Mathematics, Philadelphia, 1987), pp. 116–122
61. Y. Diouane, S. Gratton, X. Vasseur, L.N. Vicente, H. Calandra, A parallel evolution strategy for an earth imaging problem in geophysics. *Optim. Eng.* **17**(1), 3–26 (2016)
62. M. Dodangeh, L.N. Vicente, Z. Zhang, On the optimal order of worst case complexity of direct search. *Optim. Lett.* **10**(4), 699–708 (2016)
63. E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles. *Math. Program.* **91**(2), 201–213 (2002)
64. E.D. Dolan, R.M. Lewis, V. Torczon, On the local convergence of pattern search. *SIAM J. Optim.* **14**(2), 567–583 (2003)
65. D.W. Dresigmeyer, Direct search methods over Riemannian manifolds. Technical Report LA-UR-06-7416, Los Alamos National Laboratory, Los Alamos (2006)
66. D.W. Dresigmeyer, Equality constraints, Riemannian manifolds and direct search methods. Technical Report LA-UR-06-7406, Los Alamos National Laboratory, Los Alamos (2006)
67. D.W. Dresigmeyer, Direct search algorithms over Lipschitz manifolds. Technical Report LA-UR-07-1073, Los Alamos National Laboratory, Los Alamos (2007)
68. E. Fermi, N. Metropolis, Numerical solution of a minimum problem. Los Alamos Unclassified Report LA-1492, Los Alamos National Laboratory, Los Alamos (1952)
69. D.E. Finkel, C.T. Kelley, Convergence analysis of the DIRECT algorithm. Technical Report CRSC-TR04-28, Center for Research in Scientific Computation (2004)
70. D.E. Finkel, C.T. Kelley, Additive scaling and the DIRECT algorithm. *J. Glob. Optim.* **36**, 597–608 (2006)
71. D.E. Finkel, C.T. Kelley, Convergence analysis of sampling methods for perturbed Lipschitz functions. *Pac. J. Optim.* **5**(2), 339–350 (2009)
72. P.J. Fleming, R.C. Purshouse, Evolutionary algorithms in control systems engineering: a survey. *Control Eng. Pract.* **10**(11), 1223–1241 (2002)
73. R. Fletcher, S. Leyffer, Nonlinear programming without a penalty function. *Math. Program. Ser. A* **91**, 239–269 (2002)

74. R. Fletcher, N.I.M. Gould, S. Leyffer, P.L. Toint, A. Wächter, On the global convergence of trust-region SQP-filter algorithms for general nonlinear programming. *SIAM J. Optim.* **13**(3), 635–659 (2002)
75. R. Fletcher, S. Leyffer, P.L. Toint, On the global convergence of a filter—SQP algorithm. *SIAM J. Optim.* **13**(1), 44–59 (2002)
76. A. Fortin, Analyse numérique pour ingénieurs, deuxième édition. Éditions de l’École Polytechnique de Montréal (2001)
77. K.R. Fowler, C.T. Kelley, C.T. Miller, C.E. Kees, R.W. Darwin, J.P. Reese, M.W. Farthing, M.S.C. Reed, Solution of a well-field design problem with implicit filtering. *Optim. Eng.* **5**(2), 207–234 (2004)
78. A.F. Freitas, A survey of evolutionary algorithms for data mining and knowledge discovery, in *Advances in Evolutionary Computing: Theory and Applications*, ed. by A. Ghosh, S. Tsutsui (Springer, New York, 2003), pp. 819–845
79. J.M. Gablonsky, C.T. Kelley, A Locally-biased form of the DIRECT algorithm. *J. Glob. Optim.* **21**(1), 27–37 (2001)
80. U.M. García-Palomares, J.F. Rodríguez, New sequential and parallel derivative-free algorithms for unconstrained optimization. *SIAM J. Optim.* **13**(1), 79–96 (2002)
81. R. Garmanjani, L.N. Vicente, Smoothing and worst-case complexity for direct-search methods in nonsmooth optimization. *IMA J. Numer. Anal.* **33**, 1008–1028 (2013)
82. R. Garmanjani, D. Júdice, L.N. Vicente, Trust-region methods without using derivatives: worst case complexity and the nonsmooth case. *SIAM J. Optim.* **26**(4), 1987–2011 (2016)
83. S. Gill, A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proc. Camb. Philos. Soc.* **47**, 95–108 (1951)
84. P. Gilmore, T.D. Choi, O. Eslinger, C.T. Kelley, H.A. Patrick, J.M. Gablonsky, IFFCO (implicit filtering for constrained optimization). Software available at <http://www4.ncsu.edu/~ctk/iffco.html>
85. P. Gilmore, C.T. Kelly, C.T. Miller, G.A. Williams, Implicit filtering and optimal design problems, in *Optimal Design and Control*, ed. by J. Borggaard, J. Burkhardt, M. Gunzberger, J. Peterson. Progress in Systems and Control Theory, vol. 19 (Birkhäuser, Cambridge, 1995), pp. 159–176
86. R.B. Gramacy, S. Le Digabel, The mesh adaptive direct search algorithm with treed Gaussian process surrogates. *Pac. J. Optim.* **11**(3), 419–447 (2015)
87. S. Gratton, C.W. Royer, L.N. Vicente, A second-order globally convergent direct-search method and its worst-case complexity. *Optimization* **65**(6), 1105–1128 (2016)
88. G.A. Gray, T.G. Kolda, Algorithm 856: APPSPACK 4.0: asynchronous parallel pattern search for derivative-free optimization. *ACM Trans. Math. Softw.* **32**(3), 485–507 (2006)
89. J.D. Griffin, K.R. Fowler, G.A. Gray, T. Hemker, M.D. Parno, Derivative-free optimization via evolutionary algorithms guiding local search (EAGLS) for MINLP. *Pac. J. Optim.* **7**(3), 425–442 (2011)
90. W.L. Hare, Using derivative free optimization for constrained parameter selection in a home and community care forecasting model, in *International Perspectives on Operations Research and Health Care*. Proceedings of the 34th Meeting of the EURO Working Group on Operational Research Applied to Health Sciences, pp. 61–73 (2010)

91. W.L. Hare, Y. Lucet, Derivative-free optimization via proximal point methods. *J. Optim. Theory Appl.* **160**(1), 204–220 (2014)
92. W. Hare, M. Macklem, Derivative-free optimization methods for finite minimax problems. *Optim. Methods Softw.* **28**(2), 300–312 (2013)
93. W. Hare, J. Nutini, A derivative-free approximate gradient sampling algorithm for finite minimax problems. *Comput. Optim. Appl.* **56**(1), 1–38 (2013)
94. W. Hare, H. Song, On the cardinality of positively linearly independent sets. *Optim. Lett.* **10**(4), 649–654 (2016)
95. W. Hare, J. Nutini, S. Tesfamariam, A survey of non-gradient optimization methods in structural engineering. *Adv. Eng. Softw.* **59**, 19–28 (2013)
96. R.E. Hayes, F.H. Bertrand, C. Audet, S.T. Kolaczkowski, Catalytic combustion kinetics: using a direct search algorithm to evaluate kinetic parameters from light-off curves. *Can. J. Chem. Eng.* **81**(6), 1192–1199 (2003)
97. J.-B. Hiriart-Urruty, C. Lemaréchal, *Convex Analysis and Minimization Algorithms* (Springer, Berlin, 1993)
98. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (University of Michigan Press, Ann Arbor, 1975)
99. R. Hooke, T.A. Jeeves, “Direct search” solution of numerical and statistical problems. *J. Assoc. Comput. Mach.* **8**(2), 212–229 (1961)
100. H.H. Hoos, Automated algorithm configuration and parameter tuning, in *Autonomous Search*, ed. by Y. Hamadi, E. Monfroy, F. Saubion (Springer, Berlin, 2012), pp. 37–71
101. E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, A.C. Ponce Leon F. de Carvalho, A survey of evolutionary algorithms for clustering. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **39**(2), 133–155 (2009)
102. J. Jahn, *Introduction to the Theory of Nonlinear Optimization* (Springer, Berlin, 1994)
103. D.R. Jones, M. Schonlau, W.J. Welch, Efficient global optimization of expensive black box functions. *J. Glob. Optim.* **13**(4), 455–492 (1998)
104. C.T. Kelley, Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. *SIAM J. Optim.* **10**(1), 43–55 (1999)
105. C.T. Kelley, *Implicit Filtering* (Society for Industrial and Applied Mathematics, Philadelphia, 2011)
106. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth (IEEE Service Center, Piscataway, 1995)
107. S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
108. M. Kokkolaras, C. Audet, J.E. Dennis Jr., Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optim. Eng.* **2**(1), 5–29 (2001)
109. T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods. *SIAM Rev.* **45**(3), 385–482 (2003)
110. T.G. Kolda, R.M. Lewis, V. Torczon, Stationarity results for generating set search for linearly constrained optimization. *SIAM J. Optim.* **17**(4), 943–968 (2006)

111. J. Larson, M. Menickelly, S.M. Wild, Manifold sampling for  $\ell_1$  nonconvex optimization. *SIAM J. Optim.* **26**(4), 2540–2563 (2016)
112. S. Le Digabel, Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm. *ACM Trans. Math. Softw.* **37**(4), 44:1–44:15 (2011)
113. S. Le Digabel, S.M. Wild, A taxonomy of constraints in simulation-based optimization. Technical Report G-2015-57, Les cahiers du GERAD (2015)
114. E.B. Leach, A note on inverse function theorem, in *Proceedings of the American Mathematical Society*, vol. 12, pp. 694–697 (1961)
115. R.M. Lewis, V. Torczon, Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, Institute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681–2199 (1996)
116. R.M. Lewis, V. Torczon, Rank ordering and positive bases in pattern search algorithms. Technical Report TR96-71, ICASE, NASA Langley Research Center (1999)
117. R.M. Lewis, V. Torczon, M.W. Trosset, Why pattern search works. *Optima* **59**, 1–7 (1998). Also available as ICASE Technical Report 98–57. ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681–2199
118. R.M. Lewis, V. Torczon, M.W. Trosset, Direct search methods: then and now. *J. Comput. Appl. Math.* **124**(1–2), 191–207 (2000)
119. Q. Liu, J. Zeng, G. Yang, MrDIRECT: a multilevel robust DIRECT algorithm for global optimization problems. *J. Glob. Optim.* **62**(2), 205–227 (2015)
120. G. Liuzzi, S. Lucidi, F. Rinaldi, Derivative-free methods for mixed-integer constrained optimization problems. *J. Optim. Theory Appl.* **164**(3), 933–965 (2015)
121. S. Lophaven, H. Nielsen, J. Søndergaard, Dace: a matlab Kriging toolbox version 2.0. Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark (2002)
122. S. Lucidi, M. Sciandrone, On the global convergence of derivative-free methods for unconstrained optimization. *SIAM J. Optim.* **13**(1), 97–116 (2002)
123. S. Lucidi, V. Piccialli, M. Sciandrone, An algorithm model for mixed variable programming. *SIAM J. Optim.* **15**(4), 1057–1084 (2005)
124. J.M. Martínez, F.N.C. Sobral, Constrained derivative-free optimization on thin domains. *J. Glob. Optim.* **56**(3), 1217–1232 (2013)
125. K.I.M. McKinnon, Convergence of the Nelder-Mead simplex method to a non-stationary point. *SIAM J. Optim.* **9**, 148–158 (1998)
126. J.C. Meza, M.L. Martinez, On the use of direct search methods for the molecular conformation problem. *J. Comput. Chem.* **15**, 627–632 (1994)
127. J.C. Meza, R.S. Judson, T.R. Faulkner, A.M. Treasurywala, A comparison of a direct search method and a genetic algorithm for conformational searching. *J. Comput. Chem.* **17**(9), 1142–1151 (1996)
128. J.J. Moré, S.M. Wild, Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **20**(1), 172–191 (2009)
129. J. Müller, MISO: mixed-integer surrogate optimization framework. *Optim. Eng.* **17**(1), 177–203 (2016)
130. J. Müller, C.A. Shoemaker, R. Piché, SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Comput. Oper. Res.* **40**(5), 1383–1400 (2013)

131. L. Nazareth, P. Tseng, Gilding the lily: a variant of the Nelder–Mead algorithm based on golden-section search. *Comput. Optim. Appl.* **22**, 133–144 (2002)
132. J.A. Nelder, R. Mead, A simplex method for function minimization. *Comput. J.* **7**(4), 308–313 (1965)
133. J. Nocedal, S.J. Wright, *Numerical Optimization*. Springer Series in Operations Research (Springer, New York, 1999)
134. R. Oeuvray, M. Bierlaire, Boosters: a derivative-free algorithm based on radial basis functions. *Int. J. Model. Simul.* **29**(1), 26–36 (2009)
135. E. Polak, M. Wetter, Precision control for generalized pattern search algorithms with adaptive precision function evaluations. *SIAM J. Optim.* **16**(3), 650–669 (2006)
136. M. Porcelli, P.L. Toint, BFO, a trainable derivative-free Brute Force Optimizer for nonlinear bound-constrained optimization and equilibrium computations with continuous and discrete variables. *ACM Trans. Math. Softw.* Article 6, 44:1, pp. 25 (2017)
137. M.J.D. Powell, A view of unconstrained minimization algorithms that do not require derivatives. *ACM Trans. Math. Softw.* **1**(2), 97–107 (1975)
138. M.J.D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in *Advances in Optimization and Numerical Analysis*. Proceedings of the 6th Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, ed. by S. Gomez, J.-P. Hennart, vol. 275 (Kluwer Academic Publishers, Dordrecht, 1994), pp. 51–67
139. M.J.D. Powell, UOBYQA: unconstrained optimization by quadratic approximation. *Math. Program.* **92**(3), 555–582 (2002)
140. M.J.D. Powell, The NEWUOA software for unconstrained optimization without derivatives, in *Large-Scale Nonlinear Optimization*, ed. by P. Pardalos, G. Pillo, M. Roma. Nonconvex Optimization and Its Applications, vol. 83 (Springer, Berlin, 2006), pp. 255–297
141. M.J.D. Powell, The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, Cambridge University (2009)
142. C.J. Price, I.D. Coope, Frames and grids in unconstrained and linearly constrained optimization: a nonsmooth approach. *SIAM J. Optim.* **14**, 415–438 (2003)
143. A. Ralston, Runge-Kutta methods with minimum error bounds. *Math. Comput.* **16**, 431–437 (1962)
144. I. Rechenberg, Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. PhD thesis, Technische Universität Berlin, Berlin (1971)
145. R.G. Regis, The calculus of simplex gradients. *Optim. Lett.* **9**(5), 845–865 (2015)
146. R.G. Regis, Multi-objective constrained black-box optimization using radial basis function surrogates. *J. Comput. Sci.* **16**, 140–155 (2016)
147. R.G. Regis, On the convergence of adaptive stochastic search methods for constrained and multi-objective black-box optimization. *J. Optim. Theory Appl.* **170**(3), 932–959 (2016)
148. R.G. Regis, On the properties of positive spanning sets and positive bases. *Optim. Eng.* **17**(1), 229–262 (2016)

149. R.G. Regis, C.A. Shoemaker, Constrained global optimization of expensive black box functions using radial basis functions. *J. Glob. Optim.* **31**, 153–171 (2005)
150. R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, 1970)
151. R.T. Rockafellar, Generalized directional derivatives and subgradients of non-convex functions. *Can. J. Math.* **32**(2), 257–280 (1980)
152. P.R. Sampaio, P.L. Toint, A derivative-free trust-funnel method for equality-constrained nonlinear optimization. *Comput. Optim. Appl.* **61**(1), 25–49 (2015)
153. S.E. Selvan, P.B. Borckmans, A. Chattopadhyay, P.-A. Absil, Spherical mesh adaptive direct search for separating quasi-uncorrelated sources by range-based independent component analysis. *Neural Comput.* **25**(9), 2486–2522 (2013)
154. J. Søndergaard, Optimization using surrogate models — by the space mapping technique. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark (2003)
155. M.C. Spillane, E. Gica, V.V. Titov, Tsunameter network design for the U.S. DART array. AGU Fall Meeting Abstracts, p. A1368 (December 2009)
156. P. Stein, Classroom notes: a note on the volume of a simplex. *Am. Math. Mon.* **73**(3), 299–301 (1966)
157. M. Strasser, Übertragung des Optimierungsverfahrens von Nelder und Mead auf restringierte Probleme. Diploma thesis, Numerical Mathematics Group, Technical University of Darmstadt (1994)
158. The CUTER/st test problem set. <http://www.cuter.rl.ac.uk/Problems/mastsif.shtml>
159. V. Torczon, On the convergence of pattern search algorithms. *SIAM J. Optim.* **7**(1), 1–25 (1997)
160. V. Torczon, M.W. Trosset, From evolutionary operation to parallel direct search: pattern search algorithms for numerical optimization. *Comput. Sci. Stat.* **29**, 396–401 (1998)
161. M.W. Trosset, I know it when I see it: toward a definition of direct search methods. SIAG/OPT Views-and-News: Forum SIAM Activity Group Optim. **9**, 7–10 (1997)
162. P. Tseng, Fortified-descent simplicial search method: a general approach. *SIAM J. Optim.* **10**, 269–288 (1999)
163. B. Van Dyke, T.J. Asaki, Using QR decomposition to obtain a new instance of mesh adaptive direct search with uniformly distributed polling directions. *J. Optim. Theory Appl.* **159**(3), 805–821 (2013)
164. L.N. Vicente, Worst case complexity of direct search. *EURO J. Comput. Optim.* **1**(1), 143–153 (2013)
165. L.N. Vicente, A.L. Custódio, Analysis of direct searches for discontinuous functions. *Math. Program.* **133**(1–2), 299–325 (2012)
166. S.M. Wild, C.A. Shoemaker, Global convergence of radial basis function trust region derivative-free algorithms. *SIAM J. Optim.* **21**(3), 761–781 (2011)
167. S.M. Wild, R.G. Regis, C.A. Shoemaker, ORBIT: optimization by radial basis function interpolation in trust-regions. *SIAM J. Sci. Comput.* **30**(6), 3197–3219 (2008)

168. D. Winfield, Function and functional optimization by interpolation in data tables. PhD thesis, Harvard University (1969)
169. D. Winfield, Function minimization by interpolation in a data table. *J. Inst. Math. Appl.* **12**, 339–347 (1973)
170. T.A. Winslow, R.J. Trew, P. Gilmore, C.T. Kelley, Doping profiles for optimum class B performance of GaAs MESFET amplifiers, in *Proceedings IEEE/Cornell Conference on Advanced Concepts in High Speed Devices and Circuits*, pp. 188–197 (1991)
171. M.H. Wright, Direct search methods: once scorned, now respectable, in *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, ed. by D.F. Griffiths, G.A. Watson. Pitman Research Notes in Mathematics, vol. 344 (CRC Press, Boca Raton, 1996), pp. 191–208
172. M.H. Wright, Nelder, Mead, and the other simplex method. *Documenta Math. Extra Volume: Optimization Stories*, 271–276 (2012)
173. W.-C. Yu, Positive basis and a class of direct search techniques. *Sci. Sinica, Special Issue of Mathematics* **1**, 53–67 (1979)
174. A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art. *Swarm Evol. Comput.* **1**(1), 32–49 (2011)

# *Index*

- Accumulation point, 21  
Accuracy profile, *see also* Profile  
Accuracy value, 266  
Algorithm  
    Biobjective (BiObj), 254  
    Coordinate search (CS), 38  
    Exhaustive search (ES), 34  
    Generalized pattern search (GPS), 116  
    Genetic algorithm (GA), 58  
    Grid search (GS), 36  
    Latin hypercube sampling (LHS), 40  
    Mesh adaptive direct search (MADS), 139  
    Model-based descent (MBD), 186  
    Model-based trust region (MBTR), 202  
    Nelder-Mead (NM), 76  
    Surrogate management framework (SMF), 239  
    Two-phase extreme barrier (EB), 227  
Argmin, 4  
  
Barrier  
    extreme, 138, 227  
    progressive, 228  
Biobjective optimization, 248  
Blackbox optimization (BBO), 6  
Bounded set, 21  
  
Clarke subdifferential, 114  
Closed set, 21  
Closure, 21  
Compact set, 21  
Complete strategy, 44  
Cone, 107  
    hypertangent, 108  
  
Constraint types, 224  
    hidden, 224  
    relaxable, 224  
    unrelaxable, 224  
Constraint violation function, 226  
Continuity, 106  
Continuous differentiability, 17  
Controllable accuracy, 184  
Convergence plot, 267  
Convex  
    combination, 23  
    function, 24, 106  
    hull, 23  
    set, 22  
Coverage measure, 254  
Crossover, 64  
  
Data profile, *see also* Profile  
Dense subset, 33, 143  
Derivative-free optimization (DFO), 6  
Descent direction, 20, 101  
Differentiability, 17, 106  
Directional derivative  
    Clarke, 103  
    definition, 17, 102  
    generalized, 103  
Distance, 22  
Dominated points  
    biobjective optimization, 248  
    constrained optimization, 228  
Encoding, 63  
Extreme barrier, *see also* Barrier  
First-order Taylor error bound, 167

- Fitness
  - function valued, 60
  - rank, 60
- Frame, 136
- Fully linear models, 160
- Function
  - $\mathcal{C}^k$ , 17, 106
  - regular, 104
- Fundamental theorem of calculus, 166
- Generalized directional derivative, *see also* Directional derivative
- Generalized gradient, 114
- Gradient, 17
- Householder matrix, 144
- Incumbent solution, 38
- Index, *see also* Index
- Individual minima, 249
- Interpolation, 238
  - linear, 164
  - quadratic, 172
- Level set, 22
- Limit
  - inferior, 19
  - superior, 19
- Linear regression, 170, 238
- Lipschitz continuity, 18, 106
- McKinnon example, 81
- Mesh, 116, 136
- Moore-Penrose pseudo-inverse, 171
- Mutation, 66
- Nomad, 156
- Norm, 16
- Open set, 21
- Opportunistic strategy, 44
- Optimality condition
  - constrained, nonsmooth, 109
  - smooth, unconstrained, 20
  - smooth, unconstrained, convex, 24
  - unconstrained, nonsmooth, 107
  - unconstrained, smooth, 102
- Ordered strategy, 44
- Pareto front, 248
- Pareto optimality, 248
- Pareto set, 248
- Performance profile, *see also* Profile
- Poised
  - for Frobenius modelling, 175
  - for linear interpolation, 164
  - for linear regression, 170
  - for quadratic interpolation, 172
- Positive
  - basis, 97
  - linear independance, 97
  - spanning set, 97
- Profile
  - accuracy, 274
  - data, 272
  - performance, 269
- Progressive barrier, *see also* Barrier
- Projection, 114
- Reference point, 252
- Regular, 106
- Rheology problem, 9, 36, 46, 129, 148, 241
- Selection
  - elitism, 60
  - probabilistic gene, 65
  - roulette wheel, 61
  - tournament, 61
- Simplex
  - approximate diameter, 28
  - definition, 25
  - degenerate, 26
  - diameter, 27
  - gradient, 165
  - normalized volume, 27
  - volume, 26
- Single-objective subproblem, 252
- Squeeze theorem, 20
- Starting point, 40
- Stopping criteria, 39
- Surrogate, 12, 235, 236
  - recalibrate, 238
- Test set, 264
  - hidden structure, 265
  - test problems, 265, 278
- Trajectory plot, 267
- Trust region
  - subproblem, 204
- Variable types, 222
  - continuous, 6, 222
  - discrete, 222, 223
  - periodic, 222, 223