PRACTICAL MATHEMATICAL OPTIMIZATION

An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms

PRACTICAL MATHEMATICAL OPTIMIZATION

An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms

By

JAN A. SNYMAN
University of Pretoria, Pretoria, South Africa

Springer

Library of Congress Cataloging-in-Publication Data

A C.I.P. record for this book is available from the Library of Congress.

AMS Subject Classifications: 65K05, 90C30, 90-00

ISBN 0-387-24348-8 e-ISBN 0-387-24349-6 Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11161813

springeronline.com

To
Alta
my wife and friend

Contents

PREFACE xv

TABLE OF NOTATION xix

1 INTRODUCTION 1

1.1 What is mathematical optimization? 1

1.2 Objective and constraint functions 4

1.3 Basic optimization concepts 6

1.3.1 Simplest class of problems: Unconstrained one-dimensional minimization . . . 6

1.3.2 Contour representation of a function of two variables ($n = 2$) 7

1.3.3 Contour representation of constraint functions . . 10

1.3.4 Contour representations of constrained optimization problems 11

1.3.5 Simple example illustrating the formulation and solution of an optimization problem 12

1.3.6 Maximization 14

1.3.7 The special case of Linear Programming 14

1.3.8 Scaling of design variables 15

1.4 Further mathematical prerequisites 16

1.4.1 Convexity 16

1.4.2 Gradient vector of $f(\mathbf{x})$ 18

1.4.3 Hessian matrix of $f(\mathbf{x})$ 20

1.4.4 The quadratic function in $\mathbb{R}^n$ 20

1.4.5 The directional derivative of $f(\mathbf{x})$ in the direction $\mathbf{u}$ 21

1.5 Unconstrained minimization 21

1.5.1 Global and local minima; saddle points 23

1.5.2 Local characterization of the behaviour of a multi-variable function 24

1.5.3 Necessary and sufficient conditions for a strong local minimum at $\mathbf{x}^*$ 26

1.5.4 General indirect method for computing $\mathbf{x}^*$ 28

1.6 Exercises 31

2 LINE SEARCH DESCENT METHODS FOR UNCONSTRAINED MINIMIZATION 33

2.1 General line search descent algorithm for unconstrained minimization 33

2.1.1 General structure of a line search descent method . 34

2.2 One-dimensional line search 35

2.2.1 Golden section method 36

2.2.2 Powell's quadratic interpolation algorithm 38

2.2.3 Exercises 39

2.3 First order line search descent methods 40

2.3.1 The method of steepest descent 40

2.3.2 Conjugate gradient methods 43

2.4 Second order line search descent methods 49

2.4.1 Modified Newton's method 50

2.4.2 Quasi-Newton methods 51

2.5 Zero order methods and computer optimization subroutines 52

2.6 Test functions 53

3 STANDARD METHODS FOR CONSTRAINED OPTIMIZATION 57

3.1 Penalty function methods for constrained minimization . . 57

3.1.1 The penalty function formulation 58

3.1.2 Illustrative examples 58

3.1.3 Sequential unconstrained minimization technique (SUMT) 59

3.1.4 Simple example 60

3.2 Classical methods for constrained optimization problems 62

3.2.1 Equality constrained problems and the Lagrangian function 62

3.2.2 Classical approach to optimization with inequality constraints: the KKT conditions 70

3.3 Saddle point theory and duality 73

3.3.1 Saddle point theorem 73

3.3.2 Duality 74

3.3.3 Duality theorem 74

3.4 Quadratic programming 77

3.4.1 Active set of constraints 78

3.4.2 The method of Theil and Van de Panne 78

3.5 Modern methods for constrained optimization 81

3.5.1 The gradient projection method 81

3.5.2 Multiplier methods 89

3.5.3 Sequential quadratic programming (SQP) 93

4 NEW GRADIENT-BASED TRAJECTORY AND APPROXIMATION METHODS 97

4.1 Introduction 97

4.1.1 Why new algorithms? 97

4.1.2 Research at the University of Pretoria 98

4.2 The dynamic trajectory optimization method 100

4.2.1 Basic dynamic model 100

4.2.2 Basic algorithm for unconstrained problems (LFOP) 101

4.2.3 Modification for constrained problems (LFOPC) . 101

4.3 The spherical quadratic steepest descent method 105

4.3.1 Introduction 105

4.3.2 Classical steepest descent method revisited 106

4.3.3 The SQSD algorithm 107

4.3.4 Convergence of the SQSD method 108
4.3.5 Numerical results and conclusion 113
4.3.6 Test functions used for SQSD 117
4.4 The Dynamic-Q optimization algorithm 119
4.4.1 Introduction 119
4.4.2 The Dynamic-Q method 119
4.4.3 Numerical results and conclusion 123
4.5 A gradient-only line search method for conjugate gradient methods 126
4.5.1 Introduction 126
4.5.2 Formulation of optimization problem 127
4.5.3 Gradient-only line search 128
4.5.4 Conjugate gradient search directions and SUMT . 133
4.5.5 Numerical results 135
4.5.6 Conclusion 139
4.6 Global optimization using dynamic search trajectories . . 139
4.6.1 Introduction 139
4.6.2 The Snyman-Fatti trajectory method 141
4.6.3 The modified bouncing ball trajectory method . . 143
4.6.4 Global stopping criterion 145
4.6.5 Numerical results 148

5 EXAMPLE PROBLEMS **151**

5.1 Introductory examples 151

5.2 Line search descent methods 157

5.3 Standard methods for constrained optimization 170

5.3.1 Penalty function problems 170

5.3.2 The Lagrangian method applied to equality constrained problems 172

5.3.3 Solution of inequality constrained problems via auxiliary variables 181

5.3.4 Solution of inequality constrained problems via the Karush-Kuhn-Tucker conditions 184

5.3.5 Solution of constrained problems via the dual problem formulation 191

5.3.6 Quadratic programming problems 194

5.3.7 Application of the gradient projection method . . 199

5.3.8 Application of the augmented Lagrangian method 202

5.3.9 Application of the sequential quadratic programming method 203

6 **SOME THEOREMS** **207**

6.1 Characterization of functions and minima 207

6.2 Equality constrained problem 211

6.3 Karush-Kuhn-Tucker theory 215

6.4 Saddle point conditions 218

6.5 Conjugate gradient methods 223

6.6 DFP method 227

A THE SIMPLEX METHOD FOR LINEAR PROGRAMMING PROBLEMS 233

A.1 Introduction 233

A.2 Pivoting for increase in objective function 235

A.3 Example 236

A.4 The auxiliary problem for problem with infeasible origin . 238

A.5 Example of auxiliary problem solution 239

A.6 Degeneracy 241

A.7 The revised simplex method 242

A.8 An iteration of the RSM 244

Preface

It is intended that this book be used in senior- to graduate-level semester courses in optimization, as offered in mathematics, engineering, computer science and operations research departments. Hopefully this book will also be useful to practising professionals in the workplace.

The contents of the book represent the fundamental optimization material collected and used by the author, over a period of more than twenty years, in teaching Practical Mathematical Optimization to undergraduate as well as graduate engineering and science students at the University of Pretoria. The principal motivation for writing this work has not been the teaching of mathematics per se, but to equip students with the necessary fundamental optimization theory and algorithms, so as to enable them to solve practical problems in their own particular principal fields of interest, be it physics, chemistry, engineering design or business economics. The particular approach adopted here follows from the author's own personal experiences in doing research in solid-state physics and in mechanical engineering design, where he was constantly confronted by problems that can most easily and directly be solved via the judicious use of mathematical optimization techniques. This book is, however, not a collection of case studies restricted to the above-mentioned specialized research areas, but is intended to convey the basic optimization principles and algorithms to a general audience in such a way that, hopefully, the application to their own practical areas of interest will be relatively simple and straightforward.

Many excellent and more comprehensive texts on practical mathematical optimization have of course been written in the past, and I am much indebted to many of these authors for the direct and indirect influence their work has had in the writing of this monograph. In the text I have tried as far as possible to give due recognition to their contributions. Here, however, I wish to single out the excellent and possibly underrated book of D. A. Wismer and R. Chattergy (1978), which served to introduce the topic of nonlinear optimization to me many years ago, and which has more than casually influenced this work.

With so many excellent texts on the topic of mathematical optimization available, the question can justifiably be posed: Why another book and what is different here? Here I believe, for the first time in a relatively brief and introductory work, due attention is paid to certain inhibiting difficulties that can occur when fundamental and classical gradient-based algorithms are applied to real-world problems. Often students, after having mastered the basic theory and algorithms, are disappointed to find that due to real-world complications (such as the presence of noise and discontinuities in the functions, the expense of function evaluations and an excessive large number of variables), the basic algorithms they have been taught are of little value. They then discard, for example, gradient-based algorithms and resort to alternative non-fundamental methods. Here, in Chapter 4 on new gradient-based methods, developed by the author and his co-workers, the above mentioned inhibiting real-world difficulties are discussed, and it is shown how these optimization difficulties may be overcome without totally discarding the fundamental gradient-based approach.

The reader may also find the organisation of the material in this book somewhat novel. The first three chapters present the basic theory, and classical unconstrained and constrained algorithms, in a straightforward manner with almost no formal statement of theorems and presentation of proofs. Theorems are of course of importance, not only for the more mathematically inclined students, but also for practical people interested in constructing and developing new algorithms. Therefore some of the more important fundamental theorems and proofs are presented separately in Chapter 6. Where relevant, these theorems are referred to in the first three chapters. Also, in order to prevent cluttering, the presentation of the basic material in Chapters 1 to 3 is interspersed with very few worked out examples. Instead, a generous number of worked out example problems are presented separately in Chapter 5, in more or less the same order as the presentation of the corresponding theory given in Chapters 1 to 3. The separate presentation of the example problems may also be convenient for students who have to prepare for the inevitable tests and examinations. The instructor may also use these examples as models to easily formulate similar problems as additional exercises for the students, and for test purposes.

Although the emphasis of this work is intentionally almost exclusively on gradient-based methods for non-linear problems, the book will not be complete if only casual reference is made to the simplex method for solving Linear Programming (LP) problems (where of course use is also made of gradient information in the manipulation of the gradient vector $\mathbf{c}$ of the objective function, and the gradient vectors of the constraint functions contained in the matrix $\mathbf{A}$). It was therefore decided to include, as Appendix A, a short introduction to the simplex method for LP problems. This appendix introduces the simplex method along the lines given by Chvatel (1983) in his excellent treatment of the subject.

The author gratefully acknowledges the input and constructive comments of the following colleagues to different parts of this work: Nielen Stander, Albert Groenwold, Ken Craig and Danie de Kock. A special word of thanks goes to Alex Hay. Not only did he significantly contribute to the contents of Chapter 4, but he also helped with the production of most of the figures, and in the final editing of the manuscript. Thanks also to Craig Long who assisted with final corrections and to Alna van der Merwe who typed the first LaTeX draft.

Jan Snyman

Pretoria

Table of notation

$\mathbb{R}^n$	n-dimensional Euclidean (real) space
^{T}	(superscript only) transpose of a vector or matrix
$\mathbf{x}$	column vector of variables, a point in $\mathbb{R}^n$ $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$
$\in$	element in the set
$f(\mathbf{x}), f$	objective function
$\mathbf{x}^*$	local optimizer
$f(\mathbf{x}^*)$	optimum function value
$g_j(\mathbf{x}), g_j$	j^{th} inequality constraint function
$\mathbf{g}(\mathbf{x})$	vector of inequality constraint functions
$h_j(\mathbf{x}), h_j$	j^{th} equality constraint function
$\mathbf{h}(\mathbf{x})$	vector of equality constraint functions
C^1	set of continuous differentiable functions
C^2	set of continuous and twice continuous differentiable functions
$\min, \min\limits_{\mathbf{x}}$	minimize w.r.t. $\mathbf{x}$
$\mathbf{x}^0, \mathbf{x}^1, \dots$	vectors corresponding to points 0,1,...
$\{\mathbf{x} \mid \dots\}$	set of elements $\mathbf{x}$ such that ...
$\dfrac{\partial f}{\partial x_i}$	first partial derivative w.r.t. x_i
$\dfrac{\partial \mathbf{h}}{\partial x_i}$	$= [\dfrac{\partial h_1}{\partial x_i}, \dfrac{\partial h_2}{\partial x_i}, \dots, \dfrac{\partial h_r}{\partial x_i}]^T$
$\dfrac{\partial \mathbf{g}}{\partial x_i}$	$= [\dfrac{\partial g_1}{\partial x_i}, \dfrac{\partial g_2}{\partial x_i}, \dots, \dfrac{\partial g_m}{\partial x_i}]^T$
$\boldsymbol{\nabla}$	first derivative operator
$\boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{g}(\mathbf{x})$	gradient vector $= \left[\dfrac{\partial f}{\partial x_1}(\mathbf{x}), \dfrac{\partial f}{\partial x_2}(\mathbf{x}), \dots, \dfrac{\partial f}{\partial x_n}(\mathbf{x})\right]^T$ (here $\mathbf{g}$ not to be confused with the inequality constraint function vector)

$\boldsymbol{\nabla}^2$	second derivative operator (elements $\frac{\partial^2}{\partial x_i \partial x_j}$)
$\mathbf{H}(\mathbf{x}) = \boldsymbol{\nabla}^2 f(\mathbf{x})$	Hessian matrix (second derivative matrix)
$\left.\frac{df(\mathbf{x})}{d\lambda}\right\|_{\mathbf{u}}$	directional derivative at $\mathbf{x}$ in the direction $\mathbf{u}$
$\subset, \subseteq$	subset of
$\lvert\cdot\rvert$	absolute value
$\lVert\cdot\rVert$	Euclidean norm of vector
$\cong$	approximately equal
$F(\ )$	line search function
$F[,]$	first order divided difference
$F[,,]$	second order divided difference
$(\mathbf{a}, \mathbf{b})$	scalar product of vector $\mathbf{a}$ and vector $\mathbf{b}$
$\mathbf{I}$	identity matrix
θ_j	j^{th} auxiliary variable
L	Lagrangian function
λ_j	j^{th} Lagrange multiplier
$\boldsymbol{\lambda}$	vector of Lagrange multipliers
$\exists$	exists
$\Rightarrow$	implies
$\{\cdots\}$	set
$V[\mathbf{x}]$	set of constraints violated at $\mathbf{x}$
ϕ	empty set
$\mathcal{L}$	augmented Lagrange function
$\langle a \rangle$	maximum of a and zero
$\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$	$n \times r$ Jacobian matrix $= [\boldsymbol{\nabla} h_1, \boldsymbol{\nabla} h_2, \ldots, \boldsymbol{\nabla} h_r]$
$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$	$n \times m$ Jacobian matrix $= [\boldsymbol{\nabla} g_1, \boldsymbol{\nabla} g_2, \ldots, \boldsymbol{\nabla} g_m]$
s_i	slack variable
$\mathbf{s}$	vector of slack variables
D	determinant of matrix $\mathbf{A}$ of interest in $\mathbf{Ax} = \mathbf{b}$
D_j	determinant of matrix $\mathbf{A}$ with j^{th} column replaced by $\mathbf{b}$
$\lim\limits_{i \to \infty}$	limit as i tends to infinity

Chapter 1

INTRODUCTION

1.1 What is mathematical optimization?

Formally, *Mathematical Optimization* is the process of

(i) the *formulation* and

(ii) the *solution* of a constrained optimization problem of the general mathematical form:

$$\underset{\text{w.r.t. } \mathbf{x}}{\text{minimize}}\, f(\mathbf{x}), \ \ \mathbf{x} = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$$

subject to the constraints:

$$\begin{aligned} g_j(\mathbf{x}) &\leq 0, \quad j = 1,\ 2,\ \dots,\ m \\ h_j(\mathbf{x}) &= 0, \quad j = 1,\ 2,\ \dots,\ r \end{aligned} \tag{1.1}$$

where $f(\mathbf{x})$, $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ are scalar functions of the real *column vector* $\mathbf{x}$.

The continuous components x_i of $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ are called the (*design*) *variables*, $f(\mathbf{x})$ is the *objective function*, $g_j(\mathbf{x})$ denotes the respective *inequality constraint functions* and $h_j(\mathbf{x})$ the *equality constraint functions.*

The optimum vector $\mathbf{x}$ that solves problem (1.1) is denoted by $\mathbf{x}^*$ with corresponding optimum function value $f(\mathbf{x}^*)$. If no constraints are specified, the problem is called an *unconstrained* minimization problem.

Mathematical Optimization is often also called *Nonlinear Programming, Mathematical Programming* or *Numerical Optimization.* In more general terms Mathematical Optimization may be described as the science of determining the *best* solutions to mathematically defined problems, which may be models of physical reality or of manufacturing and management systems. In the first case solutions are sought that often correspond to minimum energy configurations of general structures, from molecules to suspension bridges, and are therefore of interest to Science and Engineering. In the second case commercial and financial considerations of economic importance to Society and Industry come into play, and it is required to make decisions that will ensure, for example, maximum profit or minimum cost.

The history of the Mathematical Optimization, where functions of many variables are considered, is relatively short, spanning roughly only 55 years. At the end of the 1940s the very important simplex method for solving the special class of linear programming problems was developed. Since then numerous methods for solving the general optimization problem (1.1) have been developed, tested, and successfully applied to many important problems of scientific and economic interest. There is no doubt that the advent of the computer was essential for the development of these optimization methods. However, in spite of the proliferation of optimization methods, there is no universal method for solving all optimization problems. According to Nocedal and Wright (1999): "...there are numerous algorithms, each of which is tailored to a particular type of optimization problem. It is often the user's responsibility to choose an algorithm that is appropriate for the specific application. This choice is an important one; it may determine whether the problem is solved rapidly or slowly and, indeed, whether the solution is found at all." In a similar vein Vanderplaats (1998) states that "The author of each algorithm usually has numerical examples which demonstrate the efficiency and accuracy of the method, and the unsuspecting practitioner will often invest a great deal of time and effort in programming an algorithm, only to find that it will not in fact solve the particular problem being attempted. This often leads to disenchantment with these techniques that can be avoided if the user is knowledgeable in the basic concepts of numerical optimization." With these representative and authoritative opinions in mind, and also taking into account the present author's personal experiences in developing algorithms and applying them to design problems in mechanics, this text has been written to provide a brief but unified introduction to optimization concepts and methods. In addition, an overview of a set of novel algorithms, developed by the author and his students at the University of Pretoria over the past twenty years, is also given.

The emphasis of this book is almost exclusively on gradient-based methods. This is for two reasons. (i) The author believes that the introduction to the topic of mathematical optimization is best done via the classical gradient-based approach and (ii), contrary to the current popular trend of using non-gradient methods, such as genetic algorithms (GA's), simulated annealing, particle swarm optimization and other evolutionary methods, the author is of the opinion that these search methods are, in many cases, computationally too expensive to be viable. The argument that the presence of numerical noise and multiple minima disqualify the use of gradient-based methods, and that the only way out in such cases is the use of the above mentioned non-gradient search techniques, is not necessarily true. It is the experience of the author that, through the judicious use of gradient-based methods, problems with numerical noise and multiple minima may be solved, and at a fraction of the computational cost of search techniques such as genetic algorithms. In this context Chapter 4, dealing with the new gradient-based methods developed by the author, is especially important. The presentation of the material is not overly rigorous, but hopefully correct, and should provide the necessary information to allow scientists and engineers to select appropriate optimization algorithms and to apply them successfully to their respective fields of interest.

Many excellent and more comprehensive texts on practical optimization can be found in the literature. In particular the author wishes to acknowledge the works of Wismer and Chattergy (1978), Chvatel (1983), Fletcher (1987), Bazaraa et al. (1993), Arora (1989), Haftka and Gündel (1992), Rao (1996), Vanderplaats (1998), Nocedal and Wright (1999) and Papalambros and Wilde (2000).

1.2 Objective and constraint functions

The values of the functions $f(\mathbf{x})$, $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$ at any point $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, may in practice be obtained in different ways:

(i) from *analytically* known *formulae*, e.g. $f(\mathbf{x}) = x_1^2 + 2x_2^2 + \sin x_3$;

(ii) as the *outcome* of some complicated *computational process*, e.g. $g_1(\mathbf{x}) = a(\mathbf{x}) - a_{\max}$, where $a(\mathbf{x})$ is the stress, computed by means of a finite element analysis, at some point in a structure, the design of which is specified by $\mathbf{x}$; or

(iii) from *measurements* taken of a *physical process*, e.g. $h_1(\mathbf{x}) = T(\mathbf{x}) - T_0$, where $T(\mathbf{x})$ is the temperature measured at some specified point in a reactor, and $\mathbf{x}$ is the vector of operational settings.

The first two ways of function evaluation are by far the most common. The optimization principles that apply in these cases, where computed function values are used, may be carried over directly to also be applicable to the case where the function values are obtained through physical measurements.

Much progress has been made with respect to methods for solving different classes of the general problem (1.1). Sometimes the solution may be obtained *analytically*, i.e. a closed-form solution in terms of a *formula* is obtained.

In general, especially for $n > 2$, solutions are usually obtained *numerically* by means of suitable *algorithms* (computational recipes).

Expertise in the *formulation* of appropriate optimization problems of the form (1.1), through which an optimum decision can be made, is gained from *experience*. This exercise also forms part of what is generally known as the *mathematical modelling* process. In brief, attempting to solve real-world problems via mathematical modelling requires the cyclic performance of the four steps depicted in Figure 1.1. The main steps are: 1) the observation and study of the real-world situation associated with a practical problem, 2) the abstraction of the problem by the construction of a mathematical model, that is described in terms

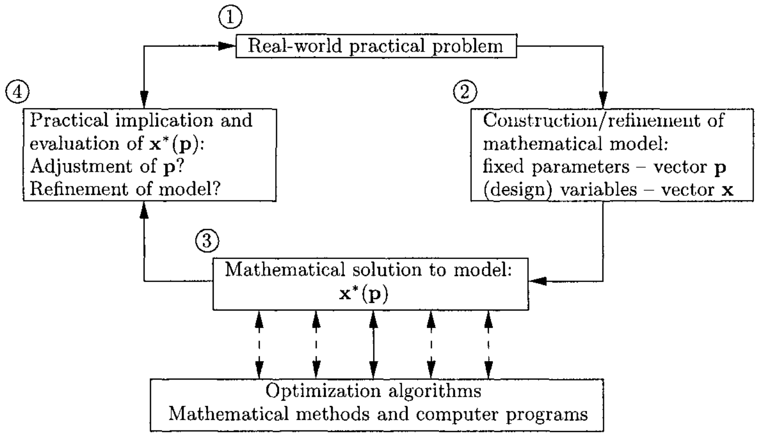

①
Real-world practical problem
②
Construction/refinement of mathematical model: fixed parameters – vector p (design) variables – vector x
③
Mathematical solution to model: x*(p)
④
Practical implication and evaluation of x*(p): Adjustment of p? Refinement of model?
Optimization algorithms Mathematical methods and computer programs

Figure 1.1: The mathematical modelling process

of preliminary fixed model parameters $\mathbf{p}$, and variables $\mathbf{x}$, the latter to be determined such that model performs in an acceptable manner, 3) the solution of a resulting purely mathematical problem, that requires an analytical or numerical parameter dependent solution $\mathbf{x}^*(\mathbf{p})$, and 4) the evaluation of the solution $\mathbf{x}^*(\mathbf{p})$ and its practical implications. After step 4) it may be necessary to adjust the parameters and refine the model, which will result in a new mathematical problem to be solved and evaluated. It may be required to perform the modelling cycle a number of times, before an acceptable solution is obtained. More often than not, the mathematical problem to be solved in 3) is a *mathematical optimization problem*, requiring a numerical solution. The *formulation* of an appropriate and consistent optimization problem (or model) is probably the most important, but unfortunately, also the *most neglected* part of Practical Mathematical Optimization.

This book gives a very brief introduction to the *formulation* of optimization problems, and deals with different *optimization algorithms* in greater depth. Since no algorithm is generally applicable to all classes of problems, the emphasis is on providing sufficient information to allow for the selection of appropriate algorithms or methods for different specific problems.

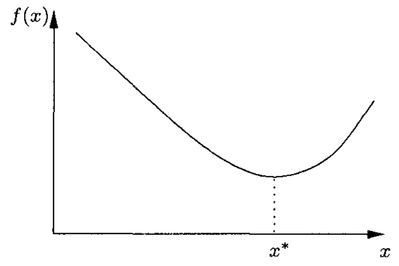


Figure 1.2: Function of single variable with optimum at x^*

1.3 Basic optimization concepts

1.3.1 Simplest class of problems: Unconstrained one-dimensional minimization

Consider the minimization of a smooth, i.e. continuous and twice continuously differentiable (C^2) function of a single real variable, i.e. the problem:

$$\underset{\text{w.r.t.}x}{\text{minimize}} f(x), \;\; x \in \mathbb{R}, \;\; f \in C^2. \tag{1.2}$$

With reference to Figure 1.2, for a strong local minimum, it is required to determine a x^* such that $f(x^*) < f(x)$ for all x.

Clearly x^* occurs where the slope is zero, i.e. where

$$f'(x) = \frac{df(x)}{dx} = 0,$$

which corresponds to the first order necessary condition. In addition *non-negative curvature* is necessary at x^*, i.e. it is required that the second order condition

$$f''(x) = \frac{d^2f(x)}{dx^2} > 0$$

must hold at x^* for a strong local minimum.

A simple *special case* is where $f(x)$ has the simple *quadratic form*:

$$f(x) = ax^2 + bx + c. \tag{1.3}$$

Since the minimum occurs where $f'(x) = 0$, it follows that the closed-form solution is given by

$$x^* = -\frac{b}{2a}, \text{ provided } f''(x^*) = a > 0. \tag{1.4}$$

If $f(x)$ has a *more general form*, then a closed-form solution is in general not possible. In this case, the solution may be obtained numerically via the *Newton-Raphson algorithm*:

Given an approximation x^0, iteratively compute:

$$x^{i+1} = x^i - \frac{f'(x^i)}{f''(x^i)}; \; i = 0,\ 1,\ 2,\ \dots \tag{1.5}$$

Hopefully $\lim_{i\to\infty} x^i = x^*$, i.e. the iterations converge, in which case a sufficiently accurate numerical solution is obtained after a finite number of iterations.

1.3.2 Contour representation of a function of two variables ($n = 2$)

Consider a function $f(\mathbf{x})$ of two variables, $\mathbf{x} = [x_1, x_2]^T$. The locus of all points satisfying $f(\mathbf{x}) = c =$ constant, forms a contour in the $x_1 - x_2$ plane. For each value of c there is a corresponding different contour.

Figure 1.3 depicts the contour representation for the example $f(\mathbf{x}) = x_1^2 + 2x_2^2$.

In three dimensions ($n = 3$), the contours are *surfaces of constant function* value. In more than three dimensions ($n > 3$) the contours are, of course, impossible to visualize. *Nevertheless, the contour representation in two-dimensional space will be used throughout the discussion of optimization techniques to help visualize the various optimization concepts.*

Other examples of 2-dimensional objective function contours are shown in Figures 1.4 to 1.6.

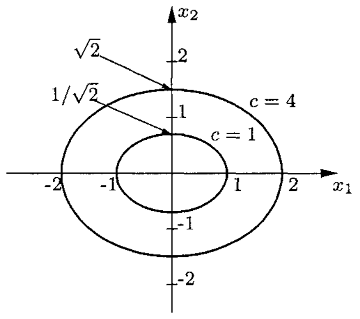


Figure 1.3: Contour representation of the function $f(\mathbf{x}) = x_1^2 + 2x_2^2$

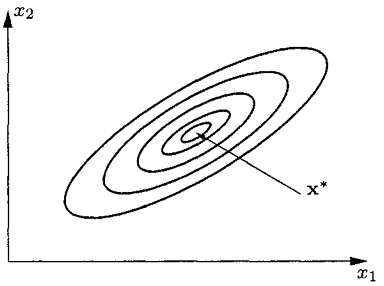


Figure 1.4: General quadratic function

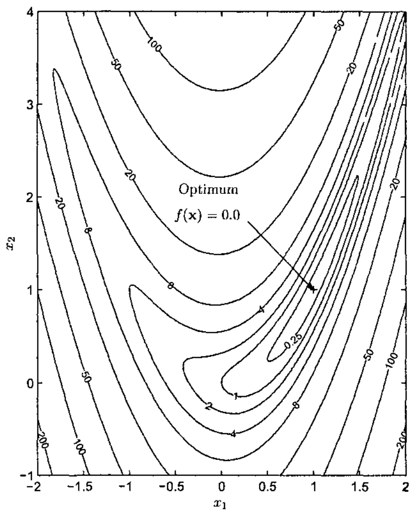


Figure 1.5: The 'banana' function $f(\mathbf{x}) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$

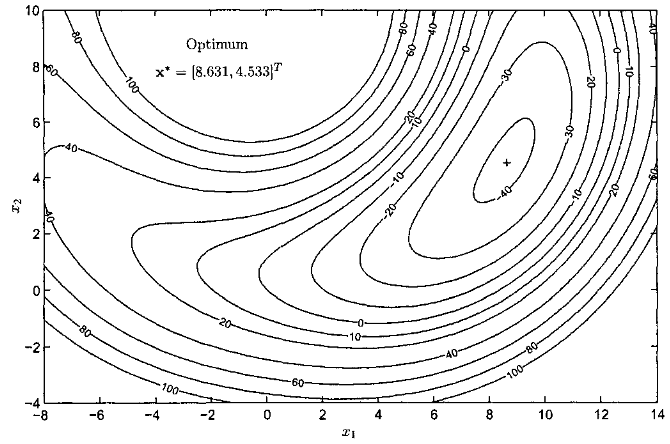


Figure 1.6: Potential energy function of a spring-force system (Vanderplaats, 1998)

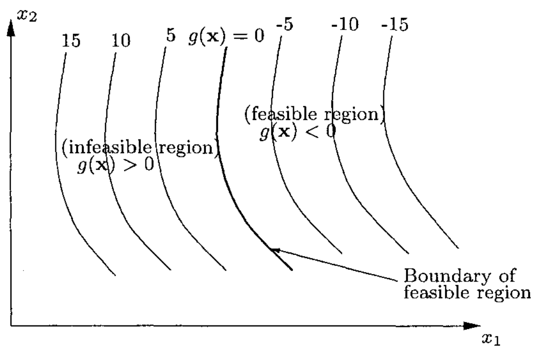


Figure 1.7: Contours within feasible and infeasible regions

1.3.3 Contour representation of constraint functions

1.3.3.1 Inequality constraint function $g(\mathrm{x})$

The contours of a typical inequality constraint function $g(\mathrm{x})$, in $g(\mathrm{x}) \leq 0$, are shown in Figure 1.7. The contour $g(\mathrm{x}) = 0$ divides the plane into a *feasible region* and an *infeasible region.*

More generally, the boundary is a surface in three dimensions and a so-called "hyper-surface" if $n > 3$, which of course cannot be visualised.

1.3.3.2 Equality constraint function $h(\mathrm{x})$

Here, as shown in Figure 1.8, only the line $h(\mathrm{x}) = 0$ is a feasible contour.

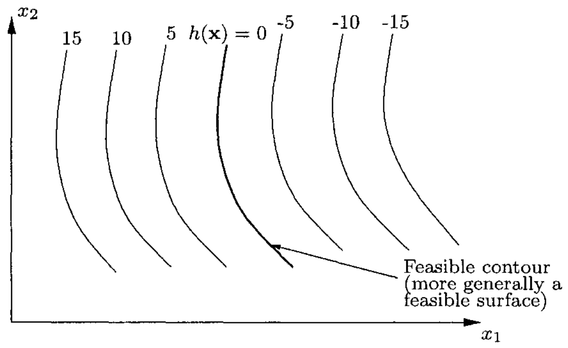
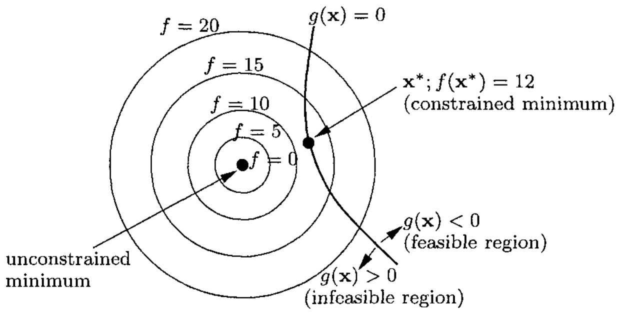


Figure 1.8: Feasible contour of equality constraint



$f = 20$
$f = 15$
$f = 10$
$f = 5$
$f = 0$
$g(\mathbf{x}) = 0$
$\mathbf{x}^*; f(\mathbf{x}^*) = 12$ (constrained minimum)
$g(\mathbf{x}) < 0$ (feasible region)
$g(\mathbf{x}) > 0$ (infeasible region)
unconstrained minimum

Figure 1.9: Contour representation of inequality constrained problem

1.3.4 Contour representations of constrained optimization problems

1.3.4.1 Representation of inequality constrained problem

Figure 1.9 graphically depicts the inequality constrained problem:

$$\begin{aligned} &\min f(\mathbf{x}) \\ \text{such that } g(\mathbf{x}) &\leq 0. \end{aligned}$$

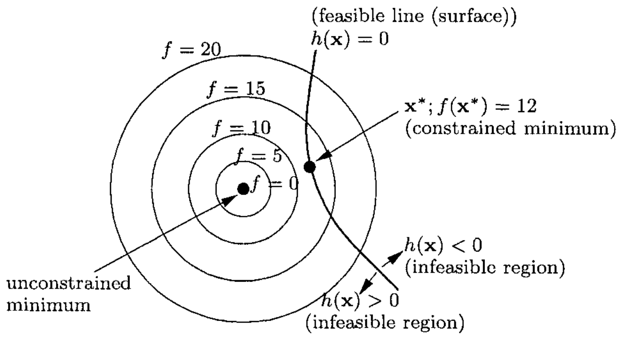


Figure 1.10: Contour representation of equality constrained problem

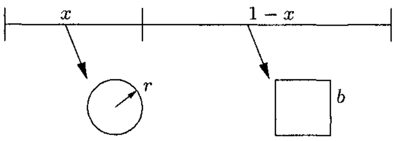


Figure 1.11: Wire divided into two pieces with $x_1 = x$ and $x_2 = 1 - x$

1.3.4.2 Representation of equality constrained problem

Figure 1.10 graphically depicts the equality constrained problem:

$$\begin{aligned} &\min f(\mathbf{x}) \\ \text{such that } h(\mathbf{x}) &= 0. \end{aligned}$$

1.3.5 Simple example illustrating the formulation and solution of an optimization problem

Problem: A length of wire 1 meter long is to be divided into two pieces, one in a circular shape and the other into a square as shown in Figure 1.11. What must the individual lengths be so that the total area is a minimum?

Formulation 1

Set length of first piece $= x$, then the area is given by $f(x) = \pi r^2 + b^2$. Since $r = \frac{x}{2\pi}$ and $b = \frac{1-x}{4}$ it follows that

$$f(x) = \pi\left(\frac{x^2}{4\pi^2}\right) + \frac{(1-x)^2}{16}.$$

The problem therefore reduces to an unconstrained minimization problem:

$$\text{minimize}\, f(x) = 0.1421x^2 - 0.125x + 0.0625.$$

Solution of Formulation 1

The function $f(x)$ is quadratic, therefore an analytical solution is given by the formula $x^* = -\frac{b}{2a}$ $(a > 0)$:

$$x^* = -\frac{-0.125}{2(0.1421)} = 0.4398 \text{ m},$$

and

$$1 - x^* = 0.5602 \text{ m with } f(x^*) = 0.0350 \text{ m}^2.$$

Formulation 2

Divide the wire into respective lengths x_1 and x_2 $(x_1 + x_2 = 1)$. The area is now given by

$$f(\mathbf{x}) = \pi r^2 + b^2 = \pi\left(\frac{x_1^2}{4\pi^2}\right) + \left(\frac{x_2}{4}\right)^2 = 0.0796x_1^2 + 0.0625x_2^2.$$

Here the problem reduces to an *equality constrained* problem:

$$\begin{aligned} \text{minimize}\, f(\mathbf{x}) &= 0.0796x_1^2 + 0.0625x_2^2 \\ \text{such that}\, h(\mathbf{x}) &= x_1 + x_2 - 1 = 0. \end{aligned}$$

Solution of Formulation 2

This constrained formulated problem is more difficult to solve. The closed-form analytical solution is not obvious and special constrained optimization techniques, such as the *method of Lagrange multipliers* to be discussed later, must be applied to solve the constrained problem analytically. The graphical solution is sketched in Figure 1.12.

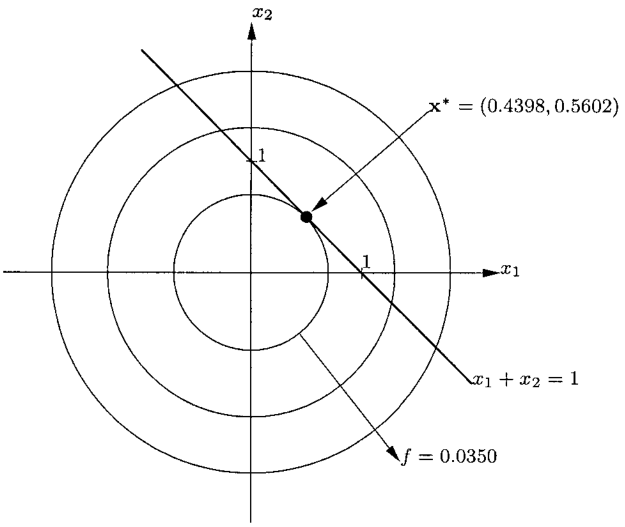


Figure 1.12: Graphical solution of Formulation 2

1.3.6 Maximization

The maximization problem: $\max\limits_{\mathbf{x}} f(\mathbf{x})$ can be cast in the standard form (1.1) by observing that $\max\limits_{\mathbf{x}} f(\mathbf{x}) = -\min\limits_{\mathbf{x}}\{-f(\mathbf{x})\}$ as shown in Figure 1.13. Therefore in applying a minimization algorithm set $F(\mathbf{x}) = -f(\mathbf{x})$.

Also if the inequality constraints are given in the non-standard form: $g_j(\mathbf{x}) \geq 0$, then set $\tilde{g}_j(\mathbf{x}) = -g_j(\mathbf{x})$. In standard form the problem then becomes:

$$\text{minimize}\, F(\mathbf{x}) \text{ such that } \tilde{g}_j(\mathbf{x}) \leq 0.$$

Once the minimizer $\mathbf{x}^*$ is obtained, the maximum value of the original maximization problem is given by $-F(\mathbf{x}^*)$.

1.3.7 The special case of Linear Programming

A very important special class of the general optimization problem arises when both the objective function and all the constraints are linear func-

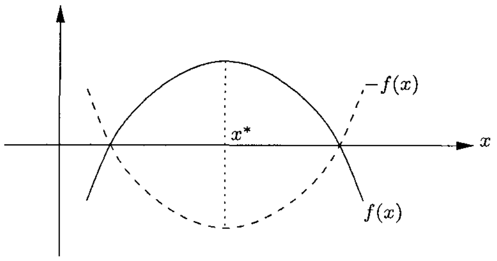


Figure 1.13: Maximization problem transformed to minimization problem

tions of $\mathbf{x}$. This is called a *Linear Programming* problem and is usually stated in the following form:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{such that} \\ &\mathbf{A}\mathbf{x} \leq \mathbf{b};\ \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{1.6}$$

where $\mathbf{c}$ is a real n-vector and $\mathbf{b}$ is a real m-vector, and $\mathbf{A}$ is a $m \times n$ real matrix. A linear programming problem in two variables is graphically depicted in Figure 1.14.

Special methods have been developed for solving linear programming problems. Of these the most famous are the simplex method proposed by Dantzig in 1947 (Dantzig, 1963) and the interior-point method (Karmarkar, 1984). A short introduction to the simplex method, according to Chvatel (1983), is given in Appendix A.

1.3.8 Scaling of design variables

In formulating mathematical optimization problems great care must be taken to ensure that the scale of the variables are more or less of the same order. If not, the formulated problem may be relatively insensitive to the variations in one or more of the variables, and any optimization algorithm will struggle to converge to the true solution, because of extreme distortion of the objective function contours as result of the poor scaling. In particular it may lead to difficulties when selecting step lengths and

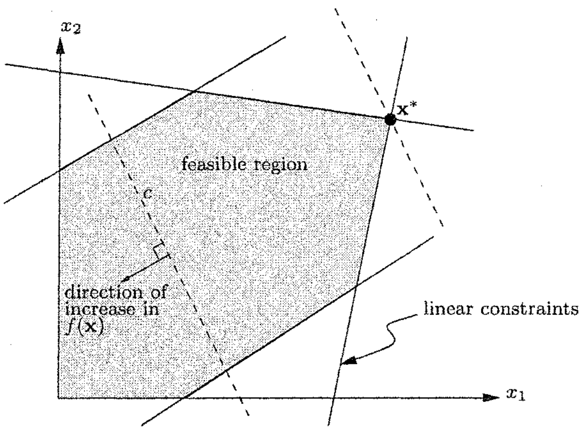


Figure 1.14: Graphical representation of a two-dimensional linear programming problem

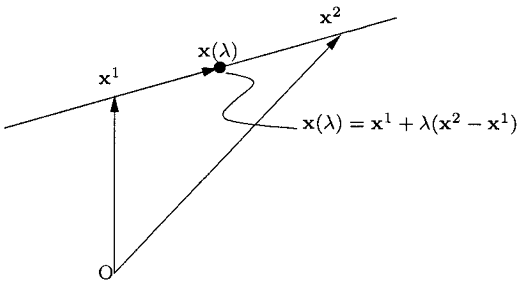
calculating numerical gradients. Scaling difficulties often occur where the variables are of different dimension and expressed in different units. Hence it is good practice, if the variable ranges are very large, to scale the variables so that all the variables will be dimensionless and vary between 0 and 1 approximately. For scaling the variables, it is necessary to establish an approximate range for each of the variables. For this, take some estimates (based on judgement and experience) for the lower and upper limits. The values of the bounds are not critical. Another related matter is the scaling or normalization of constraint functions. This becomes necessary whenever the values of the constraint functions differ by large magnitudes.

1.4 Further mathematical prerequisites

1.4.1 Convexity

A line through the points $\mathbf{x}^1$ and $\mathbf{x}^2$ in $\mathbb{R}^n$ is the set

$$L = \{\mathbf{x} \big| \mathbf{x} = \mathbf{x}^1 + \lambda(\mathbf{x}^2 - \mathbf{x}^1), \text{ for all } \lambda \in \mathbb{R}\}. \tag{1.7}$$



$\mathbf{x}^2$
$\mathbf{x}(\lambda)$
$\mathbf{x}^1$
$\mathbf{x}(\lambda) = \mathbf{x}^1 + \lambda(\mathbf{x}^2 - \mathbf{x}^1)$
O

Figure 1.15: Representation of a point on the straight line through $\mathbf{x}^1$ and $\mathbf{x}^2$

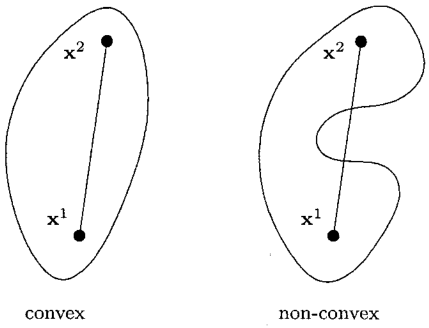


Figure 1.16: Examples of a convex and a non-convex set

Equivalently for any point $\mathbf{x}$ on the line there exists a λ such that $\mathbf{x}$ may be specified by $\mathbf{x} = \mathbf{x}(\lambda) = \lambda\mathbf{x}^2 + (1-\lambda)\mathbf{x}^1$ as shown in Figure 1.15.

1.4.1.1 Convex sets

A set X is convex if for all $\mathbf{x}^1,\ \mathbf{x}^2 \in X$ it follows that

$$\mathbf{x} = \lambda\mathbf{x}^2 + (1-\lambda)\mathbf{x}^1 \in X \text{ for all } 0 \leq \lambda \leq 1.$$

If this condition does not hold the set is non-convex (see Figure 1.16).

1.4.1.2 Convex functions

Given two points $\mathbf{x}^1$ and $\mathbf{x}^2$ in $\mathbb{R}^n$, then any point $\mathbf{x}$ on the straight line connecting them (see Figure 1.15) is given by

$$\mathbf{x} = \mathbf{x}(\lambda) = \mathbf{x}^1 + \lambda(\mathbf{x}^2 - \mathbf{x}^1),\ 0 < \lambda < 1. \tag{1.8}$$

A function $f(\mathbf{x})$ is a *convex function* over a *convex set* X if for all $\mathbf{x}^1$, $\mathbf{x}^2$ in X and for all $\lambda \in [0,1]$:

$$f(\lambda\mathbf{x}^2 + (1-\lambda)\mathbf{x}^1) \leq \lambda f(\mathbf{x}^2) + (1-\lambda)f(\mathbf{x}^1). \tag{1.9}$$

The function is strictly convex if $<$ applies. Concave functions are similarly defined.

Consider again the line connecting $\mathbf{x}^1$ and $\mathbf{x}^2$. Along this line, the function $f(\mathbf{x})$ is a function of the single variable λ:

$$F(\lambda) = f(\mathbf{x}(\lambda)) = f(\mathbf{x}^1 + \lambda(\mathbf{x}^2 - \mathbf{x}^1)). \tag{1.10}$$

This is equivalent to $F(\lambda) = f(\lambda\mathbf{x}^2 + (1-\lambda)\mathbf{x}^1)$, with $F(0) = f(\mathbf{x}^1)$ and $F(1) = f(\mathbf{x}^2)$. Therefore (1.9) may be written as

$$F(\lambda) \leq \lambda F(1) + (1-\lambda)F(0) = F_{int}$$

where F_{int} is the linearly interpolated value of F at λ as shown in Figure 1.17.

Graphically $f(\mathbf{x})$ is *convex* over the convex set X if $F(\lambda)$ has the convex form shown in Figure 1.17 for any two points $\mathbf{x}^1$ and $\mathbf{x}^2$ in X.

1.4.2 Gradient vector of $f(\mathbf{x})$

For a function $f(\mathbf{x}) \in C^2$ there exists, at any point $\mathbf{x}$ a vector of first order partial derivatives, or gradient vector:

$$\boldsymbol{\nabla} f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1}(\mathbf{x}) \\[2ex] \dfrac{\partial f}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \dfrac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix} = \mathbf{g}(\mathbf{x}). \tag{1.11}$$

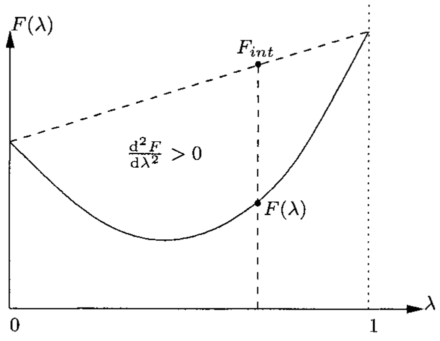


Figure 1.17: Convex form of $F(\lambda)$

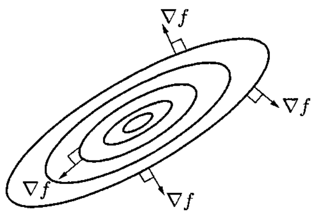


Figure 1.18: Directions of the gradient vector

It can easily be shown that if the function $f(\mathbf{x})$ is smooth, then at the point $\mathbf{x}$ the gradient vector $\boldsymbol{\nabla} f(\mathbf{x})$ (also often denoted by $\mathbf{g}(\mathbf{x})$) is always perpendicular to the contours (or surfaces of constant function value) and is in the *direction of maximum increase* of $f(\mathbf{x})$, as depicted in Figure 1.18.

1.4.3 Hessian matrix of $f(\mathbf{x})$

If $f(\mathbf{x})$ is twice continuously differentiable then at the point $\mathbf{x}$ there exists a matrix of second order partial derivatives or *Hessian matrix*:

$$\begin{aligned}\mathbf{H}(\mathbf{x}) &= \left\{\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})\right\} = \boldsymbol{\nabla}^2 f(\mathbf{x}) \qquad (1.12)\\ &= \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \dots & \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & & & \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \dots & & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}.\end{aligned}$$

Clearly $\mathbf{H}(\mathbf{x})$ is a $n \times n$ symmetrical matrix.

1.4.3.1 Test for convexity of $f(\mathbf{x})$

If $f(\mathbf{x}) \in C^2$ is defined over a convex set X, then it can be shown (see Theorem 6.1.3 in Chapter 6) that if $\mathbf{H}(\mathbf{x})$ is positive-definite for all $\mathbf{x} \in X$, then $f(\mathbf{x})$ is strictly convex over X.

To test for convexity, i.e. to determine whether $\mathbf{H}(\mathbf{x})$ is positive-definite or not, apply Sylvester's Theorem or any other suitable numerical method (Fletcher, 1987).

1.4.4 The quadratic function in $\mathbb{R}^n$

The quadratic function in n variables may be written as

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \qquad (1.13)$$

where $c \in \mathbb{R}$, $\mathbf{b}$ is a real n-vector and $\mathbf{A}$ is a $n \times n$ real matrix that can be chosen in a non-unique manner. It is usually chosen symmetrical in which case it follows that

$$\boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b};\ \mathbf{H}(\mathbf{x}) = \mathbf{A}. \qquad (1.14)$$

The function $f(\mathbf{x})$ is called positive-definite if $\mathbf{A}$ is positive-definite since, by the test in 1.4.3.1, a function $f(\mathbf{x})$ is convex if $\mathbf{H}(\mathbf{x})$ is positive-definite.

1.4.5 The directional derivative of $f(\mathbf{x})$ in the direction $\mathbf{u}$

It is usually assumed that $\|\mathbf{u}\| = 1$. Consider the differential:

$$df = \frac{\partial f}{\partial x_1} dx_1 + \cdots + \frac{\partial f}{\partial x_n} dx_n = \boldsymbol{\nabla}^T f(\mathbf{x}) d\mathbf{x}. \tag{1.15}$$

A point $\mathbf{x}$ on the line through $\mathbf{x}'$ in the direction $\mathbf{u}$ is given by $\mathbf{x} = \mathbf{x}(\lambda) = \mathbf{x}' + \lambda\mathbf{u}$, and for a small change $d\lambda$ in λ, $d\mathbf{x} = \mathbf{u}d\lambda$. Along this line $F(\lambda) = f(\mathbf{x}' + \lambda\mathbf{u})$ and the differential at any point $\mathbf{x}$ on the given line in the direction $\mathbf{u}$ is therefore given by $dF = df = \boldsymbol{\nabla}^T f(\mathbf{x})\mathbf{u}d\lambda$. It follows that the *directional derivative* at $\mathbf{x}$ in the *direction* $\mathbf{u}$ is

$$\frac{dF(\lambda)}{d\lambda} = \left.\frac{df(\mathbf{x})}{d\lambda}\right|_{\mathbf{u}} = \boldsymbol{\nabla}^T f(\mathbf{x})\mathbf{u}. \tag{1.16}$$

1.5 Unconstrained minimization

In considering the unconstrained problem: $\min_{\mathbf{x}} f(\mathbf{x}),\ \mathbf{x} \in X \subseteq \mathbb{R}^n$, the following questions arise:

(i) what are the conditions for a minimum to exist,

(ii) is the minimum unique,

(iii) are there any relative minima?

Figure 1.19 (after Farkas and Jarmai, 1997) depicts different types of minima that may arise for functions of a single variable, and for functions of two variables in the presence of equality constraints. Intuitively, with reference to Figure 1.19, one feels that a general function may have a single unique global minimum, or it may have more than one local minimum. The function may indeed have no local minimum at all, and in two dimensions the possibility of saddle points also comes to mind.

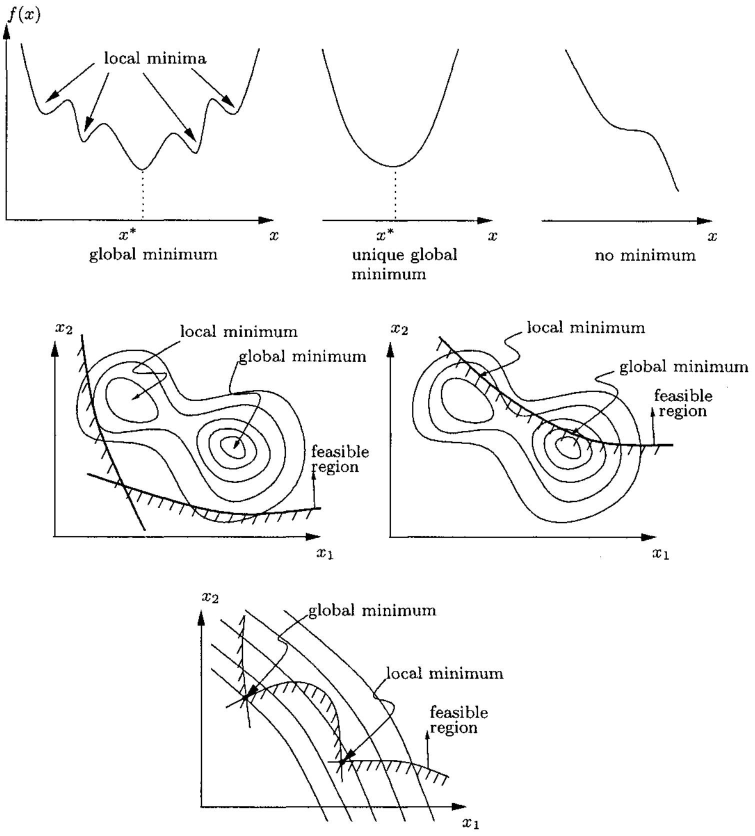


Figure 1.19: Types of minima

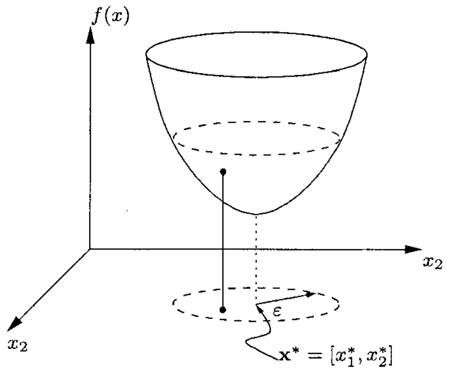


Figure 1.20: Graphical representation of the definition of a local minimum

Thus, in order to answer the above questions regarding the nature of any given function more analytically, it is necessary to give more precise meanings to the above mentioned notions.

1.5.1 Global and local minima; saddle points

1.5.1.1 Global minimum

$\mathbf{x}^*$ is a global minimum over the set X if $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x} \in X \subset \mathbb{R}^n$.

1.5.1.2 Strong local minimum

$\mathbf{x}^*$ is a strong local minimum if there exists an $\varepsilon > 0$ such that

$$f(\mathbf{x}) > f(\mathbf{x}^*) \text{ for all } \{\mathbf{x} \big| ||\mathbf{x} - \mathbf{x}^*|| < \varepsilon\}$$

where $||\cdot||$ denotes the Euclidean norm. This definition is sketched in Figure 1.20.

1.5.1.3 Test for unique local global minimum

It can be shown (see Theorems 6.1.4 and 6.1.5 in Chapter 6) that if $f(\mathbf{x})$ is strictly convex over X, then a strong local minimum is also the global minimum.

The global minimizer can be difficult to find since the knowledge of $f(\mathbf{x})$ is usually only local. Most minimization methods seek only a local minimum. An approximation to the global minimum is obtained in practice by the multi-start application of a local minimizer from randomly selected different starting points in X. The lowest value obtained after a sufficient number of trials is then taken as a good approximation to the global solution (see Snyman and Fatti, 1987; Groenwold and Snyman, 2002). If, however, it is known that the function is strictly convex over X, then only one trial is sufficient since only one local minimum, the global minimum, exists.

1.5.1.4 Saddle points

$f(\mathbf{x})$ has a saddle point at $\overline{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^0 \\ \mathbf{y}^0 \end{bmatrix}$ if there exists an $\varepsilon > 0$ such that for all $\mathbf{x}$, $\|\mathbf{x} - \mathbf{x}^0\| < \varepsilon$ and all $\mathbf{y}$, $\|\mathbf{y} - \mathbf{y}^0\| < \varepsilon$: $f(\mathbf{x}, \mathbf{y}^0) \leq f(\mathbf{x}^0, \mathbf{y}^0) \leq f(\mathbf{x}^0, \mathbf{y})$.

A contour representation of a saddle point in two dimensions is given in Figure 1.21.

1.5.2 Local characterization of the behaviour of a multi-variable function

It is assumed here that $f(\mathbf{x})$ is a smooth function, i.e., that it is a twice continuously differentiable function ($f(\mathbf{x}) \in C^2$). Consider again the line $\mathbf{x} = \mathbf{x}(\lambda) = \mathbf{x}' + \lambda \mathbf{u}$ through the point $\mathbf{x}'$ in the direction $\mathbf{u}$.

Along this line a single variable function $F(\lambda)$ may be defined:

$$F(\lambda) = f(\mathbf{x}(\lambda)) = f(\mathbf{x}' + \lambda \mathbf{u}).$$

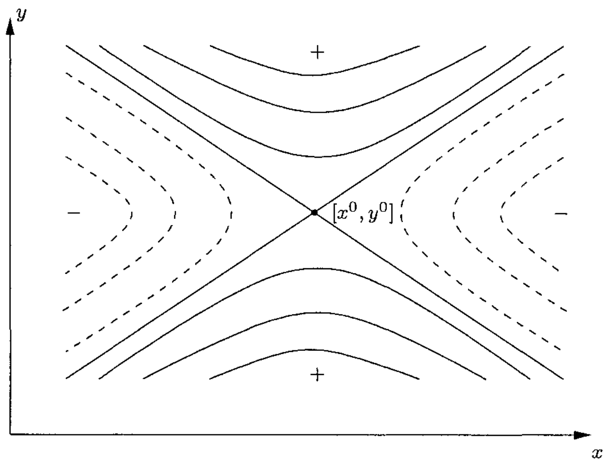


Figure 1.21: Contour representation of saddle point

It follows from (1.16) that

$$\frac{dF(\lambda)}{d\lambda} = \left.\frac{df(\mathbf{x}(\lambda))}{d\lambda}\right|_{\mathbf{u}} = \boldsymbol{\nabla}^T f(\mathbf{x}(\lambda))\mathbf{u} = g(\mathbf{x}(\lambda)) = G(\lambda)$$

which is also a single variable function of λ along the line $\mathbf{x} = \mathbf{x}(\lambda) = \mathbf{x}' + \lambda\mathbf{u}$.

Thus similarly it follows that

$$\begin{aligned}\frac{d^2F(\lambda)}{d\lambda^2} = \frac{dG(\lambda)}{d\lambda} = \left.\frac{dg(\mathbf{x}(\lambda))}{d\lambda}\right|_{\mathbf{u}} &= \boldsymbol{\nabla}^T g(\mathbf{x}(\lambda))\mathbf{u} \\ &= \boldsymbol{\nabla}^T\left(\boldsymbol{\nabla}^T f(\mathbf{x}(\lambda))\mathbf{u}\right)\mathbf{u} \\ &= \mathbf{u}^T\mathbf{H}(\mathbf{x}(\lambda))\mathbf{u}.\end{aligned}$$

Summarising: the first and second order derivatives of $F(\lambda)$ with respect to λ at any point $\mathbf{x} = \mathbf{x}(\lambda)$ on any line (any $\mathbf{u}$) through $\mathbf{x}'$ is given by

$$\frac{dF(\lambda)}{d\lambda} = \boldsymbol{\nabla}^T f(\mathbf{x}(\lambda))\mathbf{u}, \tag{1.17}$$

$$\frac{d^2F(\lambda)}{d\lambda^2} = \mathbf{u}^T\mathbf{H}(\mathbf{x}(\lambda))\mathbf{u} \tag{1.18}$$

where $\mathbf{x}(\lambda) = \mathbf{x}' + \lambda\mathbf{u}$ and $F(\lambda) = f(\mathbf{x}(\lambda)) = f(\mathbf{x}' + \lambda\mathbf{u})$.

These results may be used to obtain Taylor's expansion for a multi-variable function. Consider again the single variable function $F(\lambda)$ defined on the line through $\mathbf{x}'$ in the direction $\mathbf{u}$ by $F(\lambda) = f(\mathbf{x}' + \lambda\mathbf{u})$. It is known that the Taylor expansion of $F(\lambda)$ about 0 is given by

$$F(\lambda) = F(0) + \lambda F'(0) + \tfrac{1}{2}\lambda^2 F''(0) + \ldots \tag{1.19}$$

With $F(0) = f(\mathbf{x}')$, and substituting expressions (1.17) and (1.18) for respectively $F'(\lambda)$ and $F''(\lambda)$ at $\lambda = 0$ into (1.19) gives

$$F(\lambda) = f(\mathbf{x}' + \lambda\mathbf{u}) = f(\mathbf{x}') + \boldsymbol{\nabla}^T f(\mathbf{x}')\lambda\mathbf{u} + \tfrac{1}{2}\lambda\mathbf{u}^T\mathbf{H}(\mathbf{x}')\lambda\mathbf{u} + \ldots$$

Setting $\boldsymbol{\delta} = \lambda\mathbf{u}$ in the above gives the expansion:

$$f(\mathbf{x}' + \boldsymbol{\delta}) = f(\mathbf{x}') + \boldsymbol{\nabla}^T f(\mathbf{x}')\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{H}(\mathbf{x}')\boldsymbol{\delta} + \ldots \tag{1.20}$$

Since the above applies for any line (any $\mathbf{u}$) through $\mathbf{x}'$, it represents the general Taylor expansion for a multi-variable function about $\mathbf{x}'$. If $f(\mathbf{x})$ is fully continuously differentiable in the neighbourhood of $\mathbf{x}'$ it can be shown that the truncated second order Taylor expansion for a multi-variable function is given by

$$f(\mathbf{x}' + \boldsymbol{\delta}) = f(\mathbf{x}') + \boldsymbol{\nabla}^T f(\mathbf{x}')\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{H}(\mathbf{x}' + \theta\boldsymbol{\delta})\boldsymbol{\delta} \tag{1.21}$$

for some $\theta \in [0, 1]$. This expression is important in the analysis of the behaviour of a multi-variable function at any given point $\mathbf{x}'$.

1.5.3 Necessary and sufficient conditions for a strong local minimum at $\mathbf{x}^*$

In particular, consider $\mathbf{x}' = \mathbf{x}^*$ a strong local minimizer. Then for any line (any $\mathbf{u}$) through $\mathbf{x}'$ the behaviour of $F(\lambda)$ in a neighbourhood of $\mathbf{x}^*$ is as shown in Figure 1.22, with minimum at at $\lambda = 0$.

Clearly, a *necessary first order condition* that must apply at $\mathbf{x}^*$ (corresponding to $\lambda = 0$) is that

$$\frac{dF(0)}{d\lambda} = \boldsymbol{\nabla}^T f(\mathbf{x}^*)\mathbf{u} = 0, \quad \text{for all } \mathbf{u} \neq \mathbf{0}. \tag{1.22}$$

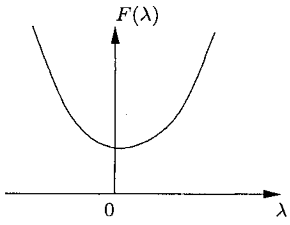


Figure 1.22: Behaviour of $F(\lambda)$ near $\lambda = 0$

It can easily be shown that this condition also implies that necessarily $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

A *necessary second order condition* that must apply at $\mathbf{x}^*$ is that

$$\frac{d^2F(0)}{d\lambda^2} = \mathbf{u}^T\mathbf{H}(\mathbf{x}^*)\mathbf{u} > 0, \quad \text{for all } \mathbf{u} \neq \mathbf{0}. \tag{1.23}$$

Conditions (1.22) and (1.23) taken together are also *sufficient conditions* (i.e. those that imply) for $\mathbf{x}^*$ to be a strong local minimum if $f(\mathbf{x})$ is continuously differentiable in the vicinity of $\mathbf{x}^*$. This can easily be shown by substituting these conditions in the Taylor expansion (1.21).

Thus in summary, the *necessary and sufficient conditions* for $\mathbf{x}^*$ to be a strong local minimum are:

$$\begin{array}{l} \nabla f(\mathbf{x}^*) = \mathbf{0} \\ \mathbf{H}(\mathbf{x}^*) \text{ positive-definite.} \end{array} \tag{1.24}$$

In the argument above it has implicitly been assumed that $\mathbf{x}^*$ is an unconstrained minimum *interior* to X. If $\mathbf{x}^*$ lies on the *boundary* of X (see Figure 1.23) then

$$\frac{dF(0)}{d\lambda} \geq 0, \text{ i.e. } \nabla^T f(\mathbf{x}^*)\mathbf{u} \geq 0 \tag{1.25}$$

for all *allowable* directions $\mathbf{u}$, i.e. for directions such that $\mathbf{x}^* + \lambda\mathbf{u} \in X$ for arbitrary small $\lambda > 0$.

Conditions (1.24) for an unconstrained strong local minimum play a very important role in the construction of practical algorithms for unconstrained optimization.

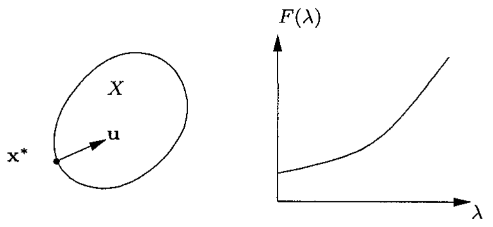


Figure 1.23: Behaviour of $F(\lambda)$ for all allowable directions of $\mathbf{u}$

1.5.3.1 Application to the quadratic function

Consider the quadratic function:

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c.$$

In this case the first order necessary condition for a minimum implies that

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}.$$

Therefore a candidate solution point is

$$\mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b}. \tag{1.26}$$

If the second order necessary condition also applies, i.e. if $\mathbf{A}$ is positive-definite, then $\mathbf{x}^*$ is a unique minimizer.

1.5.4 General indirect method for computing $\mathbf{x}^*$

The general indirect method for determining $\mathbf{x}^*$ is to solve the system of equations $\nabla f(\mathbf{x}) = \mathbf{0}$ (corresponding to the first order necessary condition in(1.24)) by some numerical method, to yield all stationary points. An obvious method for doing this is Newton's method. Since in general the system will be non-linear, multiple stationary points are possible. These stationary points must then be further analysed in order to determine whether or not they are local minima.

1.5.4.1 Solution by Newton's method

Assume $\mathbf{x}^*$ is a local minimum and $\mathbf{x}^i$ an approximate solution, with associated unknown error $\boldsymbol{\delta}$ such that $\mathbf{x}^* = \mathbf{x}^i + \boldsymbol{\delta}$. Then by applying Taylor's theorem and the first order necessary condition for a minimum at $\mathbf{x}^*$ it follows that

$$\mathbf{0} = \boldsymbol{\nabla} f(\mathbf{x}^*) = \boldsymbol{\nabla} f(\mathbf{x}^i + \boldsymbol{\delta}) = \boldsymbol{\nabla} f(\mathbf{x}^i) + \mathbf{H}(\mathbf{x}^i)\boldsymbol{\delta} + \mathrm{O}\|\boldsymbol{\delta}\|^2.$$

If $\mathbf{x}^i$ is a good approximation then $\boldsymbol{\delta} \doteq \boldsymbol{\Delta}$, the solution of the linear system $\mathbf{H}(\mathbf{x}^i)\boldsymbol{\Delta} + \boldsymbol{\nabla} f(\mathbf{x}^i) = \mathbf{0}$, obtained by ignoring the second order term in $\boldsymbol{\delta}$ above. A better approximation is therefore expected to be $\mathbf{x}^{i+1} = \mathbf{x}^i + \boldsymbol{\Delta}$ which leads to the Newton iterative scheme: Given an initial approximation $\mathbf{x}^0$, compute

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{H}^{-1}(\mathbf{x}^i)\boldsymbol{\nabla} f(\mathbf{x}^i) \tag{1.27}$$

for $i = 0,\ 1,\ 2,\ \ldots$ Hopefully $\lim\limits_{i\to\infty} \mathbf{x}^i = \mathbf{x}^*$.

1.5.4.2 Example of Newton's method applied to a quadratic problem

Consider the unconstrained problem:

$$\text{minimize}\, f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c.$$

In this case the first iteration in (1.27) yields

$$\mathbf{x}^1 = \mathbf{x}^0 - \mathbf{A}^{-1}(\mathbf{A}\mathbf{x}^0 + \mathbf{b}) = \mathbf{x}^0 - \mathbf{x}^0 - \mathbf{A}^{-1}\mathbf{b} = -\mathbf{A}^{-1}\mathbf{b}$$

i.e. $\mathbf{x}^1 = \mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b}$ in a single step (see (1.26)). This is to be expected since in this case no approximation is involved and thus $\boldsymbol{\Delta} = \boldsymbol{\delta}$.

1.5.4.3 Difficulties with Newton's method

Unfortunately, in spite of the attractive features of the Newton method, such as being quadratically convergent near the solution, the basic Newton method as described above does not always perform satisfactorily. The main difficulties are:

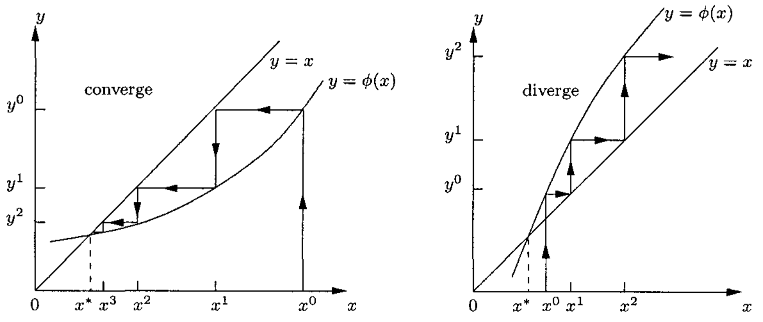


Figure 1.24: Graphical representation of Newton's iterative scheme for a single variable

(i) the method is not always convergent, even if $\mathbf{x}^0$ is close to $\mathbf{x}^*$, and

(ii) the method requires the computation of the Hessian matrix at each iteration.

The first of these difficulties may be illustrated by considering Newton's method applied to the one-dimensional problem: solve $f'(x) = 0$. In this case the iterative scheme is

$$x^{i+1} = x^i - \frac{f'(x^i)}{f''(x^i)} = \phi(x^i), \text{ for } i = 0,\ 1,\ 2,\ \ldots \tag{1.28}$$

and the solution corresponds to the fixed point x^* where $x^* = \phi(x^*)$. Unfortunately in some cases, unless x^0 is chosen to be exactly equal to x^*, convergence will not necessarily occur. In fact, convergence is dependent on the nature of the fixed point function $\phi(x)$ in the vicinity of x^*, as shown for two different ϕ functions in Figure 1.24. With reference to the graphs Newton's method is: $y^i = \phi(x^i)$, $x^{i+1} = y^i$ for $i = 0,\ 1,\ 2,\ \ldots$. Clearly in the one case where $|\phi'(x)| < 1$ convergence occurs, but in the other case where $|\phi'(x)| > 1$ the scheme diverges.

In more dimensions the situation may be even more complicated. In addition, for a large number of variables, difficulty (ii) mentioned above becomes serious in that the computation of the Hessian matrix represents a major task. If the Hessian is not available in analytical form, use can be made of automatic differentiation techniques to compute it, or it can be estimated by means of finite differences. It should also be noted that in computing the Newton step in (1.27) a $n \times n$ linear system must be solved. This represents further computational effort. Therefore in practice the simple basic Newton method is not recommended. To avoid the convergence difficulty use is made of a modified Newton method, in which a more direct search procedure is employed in the direction of the Newton step, so as to ensure descent to the minimum $\mathbf{x}^*$. The difficulty associated with the computation of the Hessian is addressed in practice through the systematic update, from iteration to iteration, of an approximation to the Hessian matrix. These improvements to the basic Newton method are dealt with in greater detail in the next chapter.

1.6 Exercises

1.6.1 Sketch the graphical solution to the following problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 2)^2$$
$$\text{such that } x_1 + 2x_2 = 4;\ x_1 \geq 0;\ x_2 \geq 0.$$

In particular indicate the feasible region: $F = \{(x_1, x_2) | x_1 + 2x_2 = 4;$
$x_1 \geq 0; x_2 \geq 0\}$ and the solution point $\mathbf{x}^*$.

1.6.2 Show that x^2 is a convex function.

1.6.3 Show that the sum of convex functions is also convex.

1.6.4 Determine the gradient vector and Hessian of the Rosenbrock function:

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

1.6.5 Write the quadratic function $f(\mathbf{x}) = x_1^2 + 2x_1x_2 + 3x_2^2$ in the standard matrix-vector notation. Is $f(\mathbf{x})$ positive-definite?

1.6.6 Write each of the following objective functions in standard form:

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c.$$

(i) $f(\mathbf{x}) = x_1^2 + 2x_1x_2 + 4x_1x_3 + 3x_2^2 + 2x_2x_3 + 5x_3^2 + 4x_1 - 2x_2 + 3x_3.$

(ii) $f(\mathbf{x}) = 5x_1^2 + 12x_1x_2 - 16x_1x_3 + 10x_2^2 - 26x_2x_3 + 17x_3^2 - 2x_1 - 4x_2 - 6x_3$.

(iii) $f(\mathbf{x}) = x_1^2 - 4x_1x_2 + 6x_1x_3 + 5x_2^2 - 10x_2x_3 + 8x_3^2$.

1.6.7 Determine the definiteness of the following quadratic form:
$f(\mathbf{x}) = x_1^2 - 4x_1x_2 + 6x_1x_3 + 5x_2^2 - 10x_2x_3 + 8x_3^2$.

Chapter 2

LINE SEARCH DESCENT METHODS FOR UNCONSTRAINED MINIMIZATION

2.1 General line search descent algorithm for unconstrained minimization

Over the last 40 years many powerful *direct search algorithms* have been developed for the unconstrained minimization of general functions. These algorithms require an initial estimate to the optimum point, denoted by $\mathbf{x}^0$. With this estimate as starting point, the algorithm generates a sequence of estimates $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \dots$, by successively searching *directly* from each point in a direction of *descent* to determine the next point. The process is terminated if either no further progress is made, or if a point $\mathbf{x}^k$ is reached (for smooth functions) at which the first necessary condition in (1.24), i.e. $\nabla f(\mathbf{x}) = \mathbf{0}$ is sufficiently accurately satisfied, in which case $\mathbf{x}^* \cong \mathbf{x}^k$. It is usually, although not always, required that the function value at the new iterate $\mathbf{x}^{i+1}$ be lower than that at $\mathbf{x}^i$.

An important sub-class of direct search methods, specifically suitable for smooth functions, are the so-called *line search* descent methods. Basic to these methods is the selection of a descent direction $\mathbf{u}^{i+1}$ at each iterate $\mathbf{x}^i$ that ensures descent at $\mathbf{x}^i$ in the direction $\mathbf{u}^{i+1}$, i.e. it is required that the directional derivative in the direction $\mathbf{u}^{i+1}$ be negative:

$$\left.\frac{df(\mathbf{x}^i)}{d\lambda}\right|_{\mathbf{u}^{i+1}} = \nabla^T f(\mathbf{x}^i)\mathbf{u}^{i+1} < 0. \tag{2.1}$$

The general structure of such descent methods is given below.

2.1.1 General structure of a line search descent method

1. Given starting point $\mathbf{x}^0$ and positive tolerances ε_1, ε_2 and ε_3, set $i = 1$.

2. Select a descent direction $\mathbf{u}^i$ (see descent condition (2.1).

3. Perform a *one-dimensional line search* in direction $\mathbf{u}^i$: i.e.

   $$\min_\lambda F(\lambda) = \min_\lambda f(\mathbf{x}^{i-1} + \lambda\mathbf{u}^i)$$

   to give minimizer λ_i.

4. Set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i\mathbf{u}^i$.

5. Test for convergence:
   if $\|\mathbf{x}^i - \mathbf{x}^{i-1}\| < \varepsilon_1$, or $\|\nabla f(\mathbf{x}^i)\| < \varepsilon_2$, or $|f(\mathbf{x}^i) - f(\mathbf{x}^{i-1})| < \varepsilon_3$, then STOP and $\mathbf{x}^* \cong \mathbf{x}^i$,

   else go to Step 6.

6. Set $i = i + 1$ and go to Step 2.

In testing for termination in step 5, a combination of the stated termination criteria may be used, i.e. instead of *or*, *and* may be specified. The structure of the above descent algorithm is depicted in Figure 2.1.

Different descent methods, within the above sub-class, differ according to the way in which the descent directions $\mathbf{u}^i$ are chosen. Another important consideration is the method by means of which the one-dimensional line search is performed.

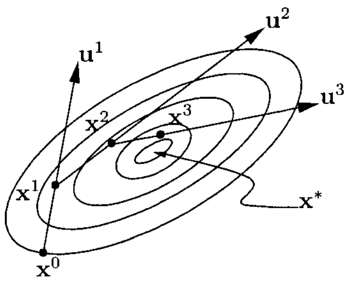


Figure 2.1: Sequence of line search descent directions and steps

2.2 One-dimensional line search

Clearly, in implementing descent algorithms of the above type, the one-dimensional minimization problem:

$$\min_{\lambda} F(\lambda), \ \lambda \in \mathbb{R} \tag{2.2}$$

is an important sub-problem. Here the minimizer is denoted by λ^*, i.e.

$$F(\lambda^*) = \min_{\lambda} F(\lambda).$$

Many one-dimensional minimization techniques have been proposed and developed over the years. These methods differ according to whether they are to be applied to smooth functions or poorly conditioned functions. For smooth functions *interpolation methods*, such as the quadratic interpolation method of Powell (1964) and the cubic interpolation algorithm of Davidon (1959), are the most efficient and accurate methods. For poorly conditioned functions, *bracketing methods*, such as the Fibonacci search method (Kiefer, 1957), which is optimal with respect to the number of function evaluations required for a prescribed accuracy, and the golden section method (Walsh, 1975), which is near optimal but much simpler and easier to implement, are preferred. Here *Powell's quadratic interpolation* method and the *golden section* method, are respectively presented as representative of the two different approaches that may be adopted to one-dimensional minimization.

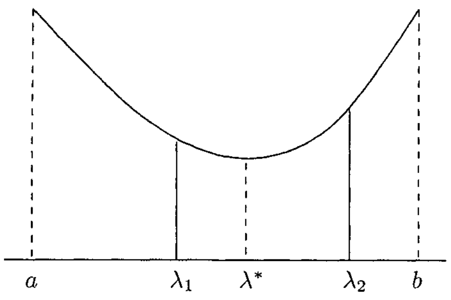


Figure 2.2: Unimodal function $F(\lambda)$ over interval $[a, b]$

2.2.1 Golden section method

It is assumed that $F(\lambda)$ is *unimodal* over the interval $[a, b]$, i.e. that it has a minimum λ^* within the interval and that $F(\lambda)$ is strictly descending for $\lambda < \lambda^*$ and strictly ascending for $\lambda > \lambda^*$, as shown in Figure 2.2.

Note that if $F(\lambda)$ is unimodal over $[a, b]$ with λ^* in $[a, b]$, then to determine a sub-unimodal interval, at least *two* evaluations of $F(\lambda)$ in $[a, b]$ must be made as indicated in Figure 2.2.

If $F(\lambda_2) > F(\lambda_1) \Rightarrow$ new unimodal interval $= [a, \lambda_2]$, and set $b = \lambda_2$ and select new λ_2; otherwise new unimodal interval $= [\lambda_1, b]$ and set $a = \lambda_1$ and select new λ_1.

Thus, the unimodal interval may successively be reduced by inspecting values of $F(\lambda_1)$ and $F(\lambda_2)$ at interior points λ_1 and λ_2.

The question arises: How can λ_1 and λ_2 be chosen in the most economic manner, i.e. such that a least number of function evaluations are required for a prescribed accuracy (i.e. for a specified uncertainty interval)? The most economic method is the Fibonacci search method. It is however a complicated method. A near optimum and more straightforward method is the golden section method. This method is a limiting form of the Fibonacci search method. Use is made of the golden ratio r when selecting the values for λ_1 and λ_2 within the unimodal interval. The value of r corresponds to the positive root of the quadratic equation: $r^2 + r - 1 = 0$, thus $r = \frac{\sqrt{5}-1}{2} = 0.618034$.

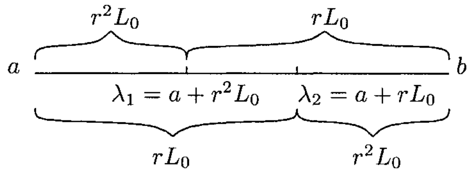


Figure 2.3: Selection of interior points λ_1 and λ_2 for golden section search

The details of the selection procedure are as follows. Given initial unimodal interval $[a, b]$ of length L_0, then choose interior points λ_1 and λ_2 as shown in Figure 2.3.

Then, if $F(\lambda_1) > F(\lambda_2) \Rightarrow$ new $[a, b] = [\lambda_1, b]$ with new interval length $L_1 = rL_0$, and

if $F(\lambda_2) > F(\lambda_1) \Rightarrow$ new $[a, b] = [a, \lambda_2]$ also with $L_1 = rL_0$.

The detailed formal algorithm is stated below.

2.2.1.1 Basic golden section algorithm

Given interval $[a, b]$ and prescribed accuracy ε; then set $i = 0$; $L_0 = b - a$, and perform the following steps:

```
1. Set $\lambda_1 = a + r^2L_0$; $\lambda_2 = a + rL_0$.
2. Compute $F(\lambda_1)$ and $F(\lambda_2)$; set $i = i + 1$.
3. If $F(\lambda_1) > F(\lambda_2)$ then
   set $a = \lambda_1$; $\lambda_1 = \lambda_2$; $L_i = (b - a)$; and $\lambda_2 = a + rL_i$,
   else
   set $b = \lambda_2$; $\lambda_2 = \lambda_1$; $L_i = (b - a)$; and $\lambda_1 = a + r^2L_i$.
4. If $L_i < \varepsilon$ then
   set $\lambda^* = \frac{b + a}{2}$; compute $F(\lambda^*)$ and STOP,
   else go to Step 2.
```

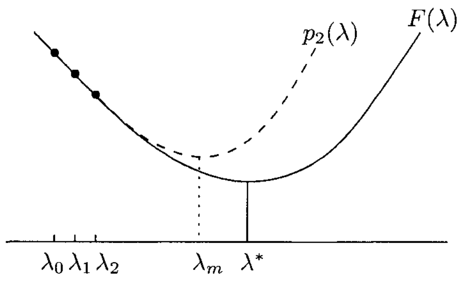


Figure 2.4: Approximate minimum λ_m via quadratic interpolation

2.2.2 Powell's quadratic interpolation algorithm

In Powell's method successive quadratic interpolation curves are fitted to function data giving a sequence of approximations to the minimum point λ^*.

With reference to Figure 2.4, the basic idea is the following. Given three data points $\{(\lambda_i, F(\lambda_i)),\ i = 1,2,3\}$, then the interpolating quadratic polynomial through these points $p_2(\lambda)$ is given by

$$p_2(\lambda) = F(\lambda_0) + F[\lambda_0,\lambda_1](\lambda-\lambda_0) + F[\lambda_0,\lambda_1,\lambda_2](\lambda-\lambda_0)(\lambda-\lambda_1) \quad (2.3)$$

where $F[\ ,\ ]$ and $F[\ ,\ ,\ ]$ respectively denote the first order and second order divided differences.

The turning point of $p_2(\lambda)$ occurs where the slope is zero, i.e. where

$$\frac{dp_2}{d\lambda} = F[\lambda_0,\lambda_1] + 2\lambda F[\lambda_0,\lambda_1,\lambda_2] - F[\lambda_0,\lambda_1,\lambda_2](\lambda_0+\lambda_1) = 0$$

which gives the turning point λ_m as

$$\lambda_m = \frac{F[\lambda_0,\lambda_1,\lambda_2](\lambda_0+\lambda_1) - F[\lambda_0,\lambda_1]}{2F[\lambda_0,\lambda_1,\lambda_2]} \cong \lambda^* \quad (2.4)$$

with the further condition that for a minimum the second derivative must be non-negative, i.e. $F[\lambda_0,\lambda_1,\lambda_2] > 0$.

The detailed formal algorithm is as follows.

2.2.2.1 Powell's interpolation algorithm

Given starting point λ_0, stepsize h, tolerance ε and maximum stepsize H; perform following steps:

1. Compute $F(\lambda_0)$ and $F(\lambda_0 + h)$.
2. *If* $F(\lambda_0) < F(\lambda_0 + h)$ evaluate $F(\lambda_0 - h)$,

   else evaluate $F(\lambda_0 + 2h)$. (The three initial values of λ so chosen constitute the initial set $(\lambda_0, \lambda_1, \lambda_2)$ with corresponding function values $F(\lambda_i)$, $i = 0, 1, 2$.)
3. Compute turning point λ_m by formula (2.4) and test for minimum or maximum.
4. *If* λ_m a minimum point *and* $|\lambda_m - \lambda_n| > H$, where λ_n is the nearest point to λ_m, then discard the point furthest from λ_m and take a step of size H from the point with lowest value in direction of descent, and go to Step 3;

   if λ_m a maximum point, then discard point nearest λ_m and take a step of size H from the point with lowest value in the direction of descent and go to Step 3;

   else continue.
5. *If* $|\lambda_m - \lambda_n| < \varepsilon$ then $F(\lambda^*) \cong \min[F(\lambda_m), F(\lambda_n)]$ and STOP,

   else continue.
6. Discard point with highest F value and replace it by λ_m; go to Step 3

Note: It is always safer to compute the next turning point by interpolation rather than by extrapolation. Therefore in Step 6: if the maximum value of F corresponds to a point which lies alone on one side of λ_m, then rather discard the point with highest value on the other side of λ_m.

2.2.3 Exercises

Apply the golden section method and Powell's method to the problems below. Compare their respective performances with regard to the number of function evaluation required to attain the prescribed accuracies.

(i) minimize $F(\lambda) = \lambda^2 + 2e^{-\lambda}$ over $[0, 2]$ with $\varepsilon = 0.01$.

(ii) maximize $F(\lambda) = \lambda \cos \lambda$ over $[0, \pi/2]$ with $\varepsilon = 0.001$.

(iii) minimize $F(\lambda) = 4(\lambda-7)/(\lambda^2+\lambda-2)$ over $[-1.9; 0.9]$ by performing only 10 function evaluations.

(iv) minimize $F(\lambda) = \lambda^4 - 20\lambda^3 + 0.1\lambda$ over $[0; 20]$ with $\varepsilon = 10^{-5}$.

2.3 First order line search descent methods

Line search descent methods (see Section 2.1.1), that use the gradient vector $\boldsymbol{\nabla} f(\mathbf{x})$ to determine the search direction for each iteration, are called *first order methods* because they employ *first order* partial derivatives of $f(\mathbf{x})$ to compute the search direction at the current iterate. The simplest and most famous of these methods is the *method of steepest descent*, first proposed by Cauchy in 1847.

2.3.1 The method of steepest descent

In this method the direction of steepest descent is used as the search direction in the line search descent algorithm given in section 2.1.1. The expression for the direction of steepest descent is derived below.

2.3.1.1 The direction of steepest descent

At $\mathbf{x}'$ we seek the unit vector $\mathbf{u}$ such that for $F(\lambda) = f(\mathbf{x}' + \lambda\mathbf{u})$, the directional derivative

$$\left.\frac{df(\mathbf{x}')}{d\lambda}\right|_{\mathbf{u}} = \frac{dF(0)}{d\lambda} = \boldsymbol{\nabla}^T f(\mathbf{x}')\mathbf{u}$$

assumes a minimum value with respect to all possible choices for the unit vector $\mathbf{u}$ at $\mathbf{x}'$ (see Figure 2.5).

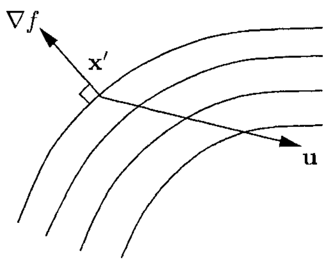


Figure 2.5: Search direction $\mathbf{u}$ relative to gradient vector at $\mathbf{x}'$

By Schwartz's inequality:

$$\boldsymbol{\nabla}^T f(\mathbf{x}')\mathbf{u} \geq -\|\boldsymbol{\nabla} f(\mathbf{x}')\|\|\mathbf{u}\| = -\|\boldsymbol{\nabla} f(\mathbf{x}')\| = \text{ least value.}$$

Clearly for the particular choice $\mathbf{u} = \dfrac{-\boldsymbol{\nabla} f(\mathbf{x}')}{\|\boldsymbol{\nabla} f(\mathbf{x}')\|}$ the directional derivative at $\mathbf{x}'$ is given by

$$\frac{dF(0)}{d\lambda} = -\boldsymbol{\nabla}^T f(\mathbf{x}')\frac{\boldsymbol{\nabla} f(\mathbf{x}')}{\|\boldsymbol{\nabla} f(\mathbf{x}')\|} = -\|\boldsymbol{\nabla} f(\mathbf{x}')\| = \text{ least value.}$$

Thus this particular choice for the unit vector corresponds to the direction of steepest descent.

The search direction

$$\mathbf{u} = \frac{-\boldsymbol{\nabla} f(\mathbf{x})}{\|\boldsymbol{\nabla} f(\mathbf{x})\|} \tag{2.5}$$

is called the *normalized steepest descent direction* at $\mathbf{x}$.

2.3.1.2 Steepest descent algorithm

Given $\mathbf{x}^0$, do for iteration $i = 1, 2, \ldots$ until convergence:

1. set $\mathbf{u}^i = \dfrac{-\boldsymbol{\nabla} f(\mathbf{x}^{i-1})}{\|\boldsymbol{\nabla} f(\mathbf{x}^{i-1})\|}$

2. set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$ where λ_i is such that

$$F(\lambda_i) = f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i) \text{ (line search).}$$

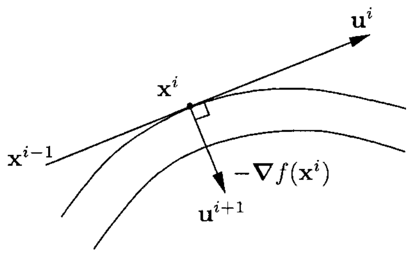


Figure 2.6: Orthogonality of successive steepest descent search directions

2.3.1.3 Characteristic property

Successive steepest descent search directions can be shown to be *orthogonal*. Consider the line search through $\mathbf{x}^{i-1}$ in the direction $\mathbf{u}^i$ to give $\mathbf{x}^i$. The condition for a minimum at λ_i, i.e. for optimal descent, is

$$\left.\frac{df(\mathbf{x}^{i-1}+\lambda_i\mathbf{u}^i)}{d\lambda}\right|_{\mathbf{u}^i} = \left.\frac{dF(\lambda_i)}{d\lambda}\right|_{\mathbf{u}^i} = \nabla^T f(\mathbf{x}^i)\mathbf{u}^i = 0$$

and with $\mathbf{u}^{i+1} = -\frac{\nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^i)\|}$ it follows that $\mathbf{u}^{i+1^T}\mathbf{u}^i = 0$ as shown in Figure 2.6.

2.3.1.4 Convergence criteria

In practice the algorithm is terminated if some convergence criterion is satisfied. Usually termination is enforced at iteration i if one, or a combination, of the following criteria is met:

(i) $\|\mathbf{x}^i - \mathbf{x}^{i-1}\| < \varepsilon_1$

(ii) $\|\nabla f(\mathbf{x}^i)\| < \varepsilon_2$

(iii) $|f(\mathbf{x}^i) - f(\mathbf{x}^{i-1})| < \varepsilon_3$.

where ε_1, ε_2 and ε_3 are prescribed small positive tolerances.

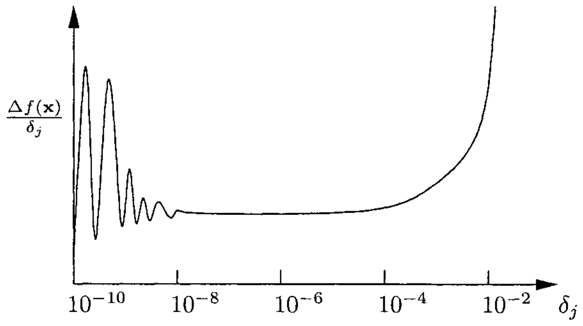


Figure 2.7: Sensitivity of finite difference approximation to δ_j

2.3.1.5 Gradients by finite differences

Often the components of the gradient vector is not analytically available in which case they may be approximated by forward finite differences:

$$\frac{\partial f(\mathbf{x})}{\partial x_j} \cong \frac{\Delta f(\mathbf{x})}{\delta_j} = \frac{f(\mathbf{x}+\boldsymbol{\delta}_j) - f(\mathbf{x})}{\delta_j} \tag{2.6}$$

where $\boldsymbol{\delta}_j = [0, 0, \dots \delta_j, 0, \dots, 0]^T$, $\delta_j > 0$ in the j-th position.

Often $\delta_j \equiv \delta$ for all $j = 1, 2, \dots, n$. A typically choice is $\delta = 10^{-6}$. If however "*numerical noise*" is present in the computation of $f(\mathbf{x})$, special care should be taken in selecting δ_j. This may require doing some numerical experiments such as, for example, determining the sensitivity of approximation (2.6) to δ_j, for each j. Typically the sensitivity graph obtained is as depicted in Figure 2.7, and for the implementation of the optimization algorithm a value for δ_j should be chosen which corresponds to a point on the plateau as shown in Figure 2.7. Better approximations, at of course greater computational expense, may be obtained through the use of central finite differences.

2.3.2 Conjugate gradient methods

In spite of its local optimal descent property the method of steepest descent often performs poorly, following a zigzagging path of ever decreas-

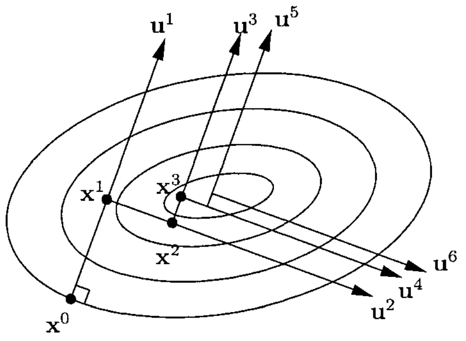


Figure 2.8: Orthogonal zigzagging behaviour of the steepest descent method

ing steps. This results in slow convergence and becomes extreme when the problem is poorly scaled, i.e. when the contours are extremely elongated. This poor performance is mainly due to the fact that the method enforces successive orthogonal search directions (see Section 2.3.1.3) as shown in Figure 2.8. Although, from a theoretical point of view, the method can be proved to be convergent, in practice the method may not effectively converge within a finite number of steps. Depending on the starting point this poor convergence also occurs when applying the method to positive-definite quadratic functions.

There is, however, a class of first order line search descent methods, known as *conjugate gradient methods*, for which it can be proved that whatever the scaling, a *method from this class will converge exactly in a finite number of iterations when applied to a positive-definite quadratic function*, i.e. to a function of the form

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{2.7}$$

where $c \in \mathbb{R}$, $\mathbf{b}$ is a real n-vector and $\mathbf{A}$ is a positive-definite $n \times n$ real symmetric matrix. Methods that have this property of *quadratic termination* are highly rated, because they are expected to also perform well on other non-quadratic functions in the neighbourhood of a local minimum. This is so, because by the Taylor expansion (1.21), it can be seen that many general differentiable functions approximate the form (2.7) near a local minimum.

2.3.2.1 Mutually conjugate directions

Two vectors $\mathbf{u}$, $\mathbf{v} \neq \mathbf{0}$ are defined to be *orthogonal* if the scalar product $\mathbf{u}^T\mathbf{v} = (\mathbf{u}, \mathbf{v}) = 0$. The concept of *mutual conjugacy* may be defined in a similar manner. Two vectors $\mathbf{u}$, $\mathbf{v} \neq \mathbf{0}$, are defined to be *mutually conjugate* with respect to the matrix $\mathbf{A}$ in (2.7) if $\mathbf{u}^T\mathbf{A}\mathbf{v} = (\mathbf{u}, \mathbf{A}\mathbf{v}) = 0$. Note that $\mathbf{A}$ is a positive definite symmetric matrix.

It can also be shown (see Theorem 6.5.1 in Chapter 6) that if the set of vectors $\mathbf{u}^i$, $i = 1, 2, \dots, n$ are *mutually conjugate*, then they form a *basis* in $\mathbb{R}^n$, i.e. any $\mathbf{x} \in \mathbb{R}^n$ may be expressed as

$$\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{u}^i \tag{2.8}$$

where

$$\lambda_i = \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x})}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)}. \tag{2.9}$$

2.3.2.2 Convergence theorem for mutually conjugate directions

Suppose $\mathbf{u}^i$, $i = 1, 2, \dots, n$ are mutually conjugate with respect to positive-definite $\mathbf{A}$, then the optimal line search descent method in Section 2.1.1, using $\mathbf{u}^i$ as search directions, converges to the unique minimum $\mathbf{x}^*$ of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$ in less than or equal to n steps.

Proof:

If $\mathbf{x}^0$ the starting point, then after i iterations:

$$\begin{aligned} \mathbf{x}^i &= \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i = \mathbf{x}^{i-2} + \lambda_{i-1}\mathbf{u}^{i-1} + \lambda_i \mathbf{u}^i = \dots \\ &= \mathbf{x}^0 + \sum_{k=1}^{i} \lambda_k \mathbf{u}^k. \end{aligned} \tag{2.10}$$

The condition for *optimal descent* at iteration i is

$$\begin{aligned} \frac{dF(\lambda_i)}{d\lambda} = \left.\frac{df(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i)}{d\lambda}\right|_{\mathbf{u}^i} &= [\mathbf{u}^i, \nabla f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i)] = 0 \\ &= [\mathbf{u}^i, \nabla f(\mathbf{x}^i)] = 0 \end{aligned}$$

i.e.

$$\begin{aligned} 0 &= (\mathbf{u}^i, \mathbf{A}\mathbf{x}^i + \mathbf{b}) \\ &= \left(\mathbf{u}^i, \mathbf{A}\left(\mathbf{x}^0 + \sum_{k=1}^{i} \lambda_k \mathbf{u}^k\right) + \mathbf{b}\right) = (\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b}) + \lambda_i(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i) \end{aligned}$$

because $\mathbf{u}^i$, $i = 1, 2, \dots, n$ are mutually conjugate, and thus

$$\lambda_i = -(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b})/(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i). \tag{2.11}$$

Substituting (2.11) into (2.10) above gives

$$\begin{aligned} \mathbf{x}^n &= \mathbf{x}^0 + \sum_{i=1}^{n} \lambda_i \mathbf{u}^i = \mathbf{x}^0 - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b})\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)} \\ &= \mathbf{x}^0 - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0)\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)} - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}(\mathbf{A}^{-1}\mathbf{b}))\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)}. \end{aligned}$$

Now by utilizing (2.8) and (2.9) it follows that

$$\mathbf{x}^n = \mathbf{x}^0 - \mathbf{x}^0 - \mathbf{A}^{-1}\mathbf{b} = -\mathbf{A}^{-1}\mathbf{b} = \mathbf{x}^*.$$

The implication of the above theorem for the case $n = 2$ and where mutually conjugate line search directions $\mathbf{u}^1$ and $\mathbf{u}^2$ are used, is depicted in Figure 2.9.

2.3.2.3 Determination of mutually conjugate directions

How can mutually conjugate search directions be found? One way is to determine all the eigenvectors $\mathbf{u}^i$, $i = 1, 2, \dots, n$ of $\mathbf{A}$. For $\mathbf{A}$ positive-definite, all the eigenvectors are mutually orthogonal and since $\mathbf{A}\mathbf{u}^i = \mu_i \mathbf{u}^i$ where μ_i is the associated eigenvalue, it follows directly that for all $i \neq j$ that $(\mathbf{u}^i, \mathbf{A}\mathbf{u}^j) = (\mathbf{u}^i, \mu_j \mathbf{u}^j) = \mu_j(\mathbf{u}^i, \mathbf{u}^j) = 0$, i.e. the eigenvectors are mutually conjugate with respect to $\mathbf{A}$. It is, however, not very practical to determine mutually conjugate directions by finding all the eigenvectors of $\mathbf{A}$, since the latter task in itself represents a computational problem of magnitude equal to that of solving the original unconstrained optimization problem via any other numerical algorithm. An easier method for obtaining mutually conjugate directions, is by means of the Fletcher-Reeves formulae (Fletcher and Reeves, 1964).

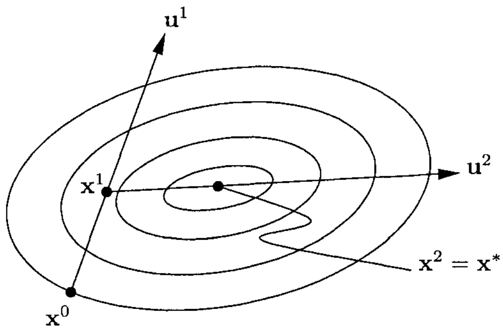


Figure 2.9: Quadratic termination of the conjugate gradient method in two steps for the case $n = 2$

2.3.2.4 The Fletcher-Reeves directions

The *Fletcher-Reeves directions* $\mathbf{u}^i$, $i = 1, 2, \dots, n$, that are listed below, can be shown (see Theorem 6.5.3 in Chapter 6) to be mutually conjugate with respect to the matrix $\mathbf{A}$ in the expression for the quadratic function in (2.7) (for which $\nabla f(\mathbf{x}) = \mathbf{Ax} + \mathbf{b}$). The explicit directions are:

$$\mathbf{u}^1 = -\nabla f(\mathbf{x}^0)$$

and for $i = 1, 2, \dots, n-1$

$$\mathbf{u}^{i+1} = -\nabla f(\mathbf{x}^i) + \beta_i \mathbf{u}^i \tag{2.12}$$

where $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$, and λ_i corresponds to the optimal descent step in iteration i, and

$$\beta_i = \frac{\|\nabla f(\mathbf{x}^i)\|^2}{\|\nabla f(\mathbf{x}^{i-1})\|^2}. \tag{2.13}$$

The *Polak-Ribiere* directions are obtained if, instead of using (2.13), β_i is computed using

$$\beta_i = \frac{(\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))^T \nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^{i-1})\|^2}. \tag{2.14}$$

If $f(\mathbf{x})$ is quadratic it can be shown (Fletcher, 1987) that (2.14) is equivalent to (2.13).

2.3.2.5 Formal Fletcher-Reeves conjugate gradient algorithm for general functions

Given $\mathbf{x}^0$ perform the following steps:

1. Compute $\nabla f(\mathbf{x}^0)$ and set $\mathbf{u}^1 = -\nabla f(\mathbf{x}^0)$.
2. For $i = 1, 2, \dots, n$ do:

   2.1 set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$ where λ_i such that

   $$f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i) \text{ (line search)},$$

   2.2 compute $\nabla f(\mathbf{x}^i)$,

   2.3 *if* convergence criteria satisfied, then STOP and $\mathbf{x}^* \cong \mathbf{x}^i$, *else* go to Step 2.4.

   2.4 *if* $1 \leq i \leq n-1$, $\mathbf{u}^{i+1} = -\nabla f(\mathbf{x}^i) + \beta_i \mathbf{u}^i$ with β_i given by (2.13).
3. Set $\mathbf{x}^0 = \mathbf{x}^n$ and go to Step 2 (restart).

If β_i is computed by (2.14) instead of (2.13) the method is known as the *Polak-Ribiere* method.

2.3.2.6 Simple illustrative example

Apply the Fletcher-Reeves method to minimize

$$f(\mathbf{x}) = \tfrac{1}{2}x_1^2 + x_1x_2 + x_2^2$$

with $\mathbf{x}^0 = [10, -5]^T$.

Solution:

Iteration 1:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \text{ and therefore } \mathbf{u}^1 = -\nabla f(\mathbf{x}^0) = \begin{bmatrix} -5 \\ 0 \end{bmatrix}.$$

$$\mathbf{x}^1 = \mathbf{x}^0 + \lambda \mathbf{u}^1 = \begin{bmatrix} 10 - 5\lambda \\ -5 \end{bmatrix} \text{ and}$$

$F(\lambda) = f(\mathbf{x}^0 + \lambda\mathbf{u}^1) = \frac{1}{2}(10 - 5\lambda)^2 + (10 - 5\lambda)(-5) + 25.$

For optimal descent

$\frac{dF}{d\lambda}(\lambda) = \left.\frac{df}{d\lambda}\right|_{\mathbf{u}^1} = -5(10 - 5\lambda) + 25 = 0$ (line search).

This gives $\lambda_1 = 1$, $\mathbf{x}^1 = \begin{bmatrix} 5 \\ -5 \end{bmatrix}$ and $\nabla f(\mathbf{x}^1) = \begin{bmatrix} 0 \\ -5 \end{bmatrix}$.

Iteration 2:

$$\mathbf{u}^2 = -\nabla f(\mathbf{x}^1) + \frac{\|\nabla f(\mathbf{x}^1)\|^2}{\|\nabla f(\mathbf{x}^0)\|^2}\mathbf{u}^1 = -\begin{bmatrix} 0 \\ -5 \end{bmatrix} + \frac{25}{25}\begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \end{bmatrix}.$$

$$\mathbf{x}^2 = \mathbf{x}^1 + \lambda\mathbf{u}^2 = \begin{bmatrix} 5 \\ -5 \end{bmatrix} + \lambda\begin{bmatrix} -5 \\ 5 \end{bmatrix} = \begin{bmatrix} 5(1-\lambda) \\ -5(1-\lambda) \end{bmatrix} \text{ and}$$

$F(\lambda) = f(\mathbf{x}^1 + \lambda\mathbf{u}^2) = \frac{1}{2}[25(1-\lambda)^2 - 50(1-\lambda)^2 + 50(1-\lambda)^2].$

Again for optimal descent

$\frac{dF}{d\lambda}(\lambda) = \left.\frac{df}{d\lambda}\right|_{\mathbf{u}^2} = -25(1-\lambda) = 0$ (line search).

This gives $\lambda_2 = 1$, $\mathbf{x}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\nabla f(\mathbf{x}^2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Therefore STOP.

The two iteration steps are shown in Figure 2.10.

2.4 Second order line search descent methods

These methods are based on Newton's method (see Section 1.5.4.1) for solving $\nabla f(\mathbf{x}) = \mathbf{0}$ iteratively: Given $\mathbf{x}^0$, then

$$\mathbf{x}^i = \mathbf{x}^{i-1} - \mathbf{H}^{-1}(\mathbf{x}^{i-1})\nabla f(\mathbf{x}^{i-1}),\ i = 1, 2, \ldots \tag{2.15}$$

As stated in Chapter 1, the main characteristics of this method are:

1. In the neighbourhood of the solution it may converge very fast, in

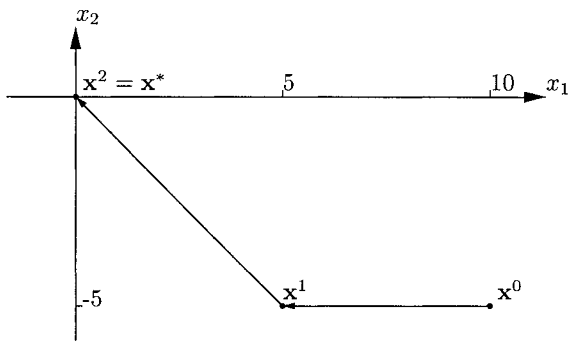


Figure 2.10: Convergence of Fletcher-Reeves method for illustrative example

fact it has the very desirous property of being quadratically convergent if it converges. Unfortunately convergence is not guaranteed and it may sometimes diverge, even from close to the solution.

2. The implementation of the method requires that $\mathbf{H}(\mathbf{x})$ be evaluated at each step.

3. To obtain the Newton step, $\boldsymbol{\Delta} = \mathbf{x}^i - \mathbf{x}^{i-1}$ it is also necessary to solve a $n \times n$ linear system $\mathbf{H}(\mathbf{x})\boldsymbol{\Delta} = -\boldsymbol{\nabla} f(\mathbf{x})$ at each iteration. This is computationally very expensive for large n, since an order n^3 multiplication operations are required to solve the system numerically.

2.4.1 Modified Newton's method

To avoid the problem of convergence (point 1. above), the computed Newton step $\boldsymbol{\Delta}$ is rather used as a search direction in the general line search descent algorithm given in Section 2.1.1. Thus at iteration i: select $\mathbf{u}^i = \boldsymbol{\Delta} = -\mathbf{H}^{-1}(\mathbf{x}^{i-1})\boldsymbol{\nabla} f(\mathbf{x}^{i-1})$, and minimize in that direction to obtain a λ_i such that

$$f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)$$

and then set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$.

2.4.2 Quasi-Newton methods

To avoid the above mentioned computational problems (2. and 3.), methods have been developed in which approximations to $\mathbf{H}^{-1}$ are applied at each iteration. Starting with an approximation $\mathbf{G}_0$ to $\mathbf{H}^{-1}$ for the first iteration, the approximation is updated after each line search. An example of such a method is the Davidon-Fletcher-Powell (DFP) method.

2.4.2.1 DFP quasi-Newton method

The structure of this (rank-1 update) method (Fletcher, 1987) is as follows.

1. Choose $\mathbf{x}^0$ and set $\mathbf{G}_0 = \mathbf{I}$.
2. Do for iteration $i = 1, 2, \dots, n$:
   - 2.1 set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$, where $\mathbf{u}^i = -\mathbf{G}_{i-1}\nabla f(\mathbf{x}^{i-1})$ and λ_i is such that $f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min\limits_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)$, $\lambda_i \geq 0$ (line search),
   - 2.2 if stopping criteria satisfied then STOP, $\mathbf{x}^* \cong \mathbf{x}^i$,
   - 2.3 set $\mathbf{v}^i = \lambda_i \mathbf{u}^i$ and
     set $\mathbf{y}^i = \nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1})$,
   - 2.4 set
     $$\mathbf{G}_i = \mathbf{G}_{i-1} + \mathbf{A}_i + \mathbf{B}_i \text{ (rank 1-update)} \tag{2.16}$$
     where $\mathbf{A}_i = \dfrac{\mathbf{v}^i \mathbf{v}^{iT}}{\mathbf{v}^{iT}\mathbf{y}^i}$, $\mathbf{B}_i = \dfrac{-\mathbf{G}_{i-1}\mathbf{y}^i(\mathbf{G}_{i-1}\mathbf{y}^i)^T}{\mathbf{y}^{iT}\mathbf{G}_{i-1}\mathbf{y}^i}$.
3. Set $\mathbf{x}^0 = \mathbf{x}^n$; $\mathbf{G}_0 = \mathbf{G}_n$ (or $\mathbf{G}_0 = \mathbf{I}$), and go to Step 2 (restart).

2.4.2.2 Characteristics of DFP method

1. The method does not require the evaluation of $\mathbf{H}$ or the explicit solution of a linear system.

2. If $\mathbf{G}_{i-1}$ is positive-definite then so is $\mathbf{G}_i$ (see Theorem 6.6.1).

3. If $\mathbf{G}_i$ is positive-definite then descent is ensured at $\mathbf{x}^i$ because

$$\begin{aligned}\left.\frac{df(\mathbf{x}^i)}{d\lambda}\right|_{\mathbf{u}^{i+1}} &= \boldsymbol{\nabla}^T f(\mathbf{x}^i)\mathbf{u}^{i+1}\\ &= -\boldsymbol{\nabla}^T f(\mathbf{x}^i)\mathbf{G}_i\boldsymbol{\nabla} f(\mathbf{x}^i) < 0, \text{ for all } \boldsymbol{\nabla} f(\mathbf{x}) \neq \mathbf{0}.\end{aligned}$$

4. The directions $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ are mutually conjugate for a quadratic function with $\mathbf{A}$ positive-definite (see Theorem 6.6.2). The method therefore possesses the desirable property of *quadratic termination* (see Section 2.3.2).

5. For quadratic functions: $\mathbf{G}_n = \mathbf{A}^{-1}$ (see again Theorem 6.6.2).

2.4.2.3 The BFGS method

The state-of-the-art quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method developed during the early 1970s (see Fletcher, 1987). This method uses a more complicated rank-2 update formula for $\mathbf{H}^{-1}$. For this method the update formula to be used in Step 2.4 of the algorithm given in Section 2.4.2.1 becomes

$$\begin{aligned}\mathbf{G}_i = \mathbf{G}_{i-1} &+ \left[1 + \frac{\mathbf{y}^{iT}\mathbf{G}_{i-1}\mathbf{y}^i}{\mathbf{v}^{iT}\mathbf{y}^i}\right]\left[\frac{\mathbf{v}^i\mathbf{v}^{iT}}{\mathbf{v}^{iT}\mathbf{y}^i}\right]\\ &- \left[\frac{\mathbf{v}^i\mathbf{y}^{iT}\mathbf{G}_{i-1} + \mathbf{G}_{i-1}\mathbf{y}^i\mathbf{v}^{iT}}{\mathbf{v}^{iT}\mathbf{y}^i}\right].\end{aligned} \tag{2.17}$$

2.5 Zero order methods and computer optimization subroutines

This chapter would not be complete without mentioning something about the large number of so-called *zero order* methods that have been developed. These methods are called such because they do not use either first order or second order derivative information, but only function values, i.e. only zero order derivative information.

Zero order methods are of the earliest methods and many of them are based on rough and ready ideas without very much theoretical background. Although these ad hoc methods are, as one may expect, much slower and computationally much more expensive than the higher order methods, they are usually reliable and easy to program. One of the most successful of these methods is the *simplex method* of Nelder and Mead (1965). This method should not be confused with the simplex method for linear programming. Another very powerful and popular method that only uses function values is the multi-variable method of Powell (1964). This method generates mutually conjugate directions by performing sequences of line searches in which only function evaluations are used. For this method Theorem 2.3.2.2 applies and the method therefore possesses the property if quadratic termination.

Amongst the more recently proposed and modern zero order methods, the method of *simulated annealing* and the so-called *genetic algorithms* (GAs) are the most prominent (see for example, Haftka and Gündel, 1992).

Computer programs are commercially available for all the unconstrained optimization methods presented in this chapter. Most of the algorithms may, for example, be found in the *Matlab Optimization Toolbox* and in the *IMSL* and *NAG* mathematical subroutine libraries.

2.6 Test functions

The efficiency of an algorithm is studied using standard functions with standard starting points $\mathbf{x}^0$. The total number of functions evaluations required to find $\mathbf{x}^*$ is usually taken as a measure of the efficiency of the algorithm. Some classical test functions (Rao, 1996) are listed below.

1. Rosenbrock's parabolic valley:

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2;\ \mathbf{x}^0 = \begin{bmatrix} -1.2 \\ 1.0 \end{bmatrix}\ \mathbf{x}^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

2. Quadratic function:

$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2;\ \mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\ \mathbf{x}^* = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

3. Powell's quartic function:

$$\begin{aligned} f(\mathbf{x}) &= (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4; \\ \mathbf{x}^0 &= [3, -1, 0, 1]^T; \ \ \mathbf{x}^* \ = \ [0, 0, 0, 0]^T. \end{aligned}$$

4. Fletcher and Powell's helical valley:

$$\begin{aligned} f(\mathbf{x}) &= 100\left((x_3 - 10\theta(x_1, x_2))^2 \right. \\ &\quad \left. + \left(\sqrt{x_1^2 + x_2^2} - 1\right)^2\right) + x_3^2; \\ \text{where } 2\pi\theta(x_1, x_2) &= \begin{cases} \arctan \dfrac{x_2}{x_1} \text{ if } x_1 > 0 \\ \pi + \arctan \dfrac{x_2}{x_1} \text{ if } x_1 < 0 \end{cases} \\ \mathbf{x}^0 &= [-1, 0, 0]^T; \ \ \mathbf{x}^* \ = \ [1, 0, 0]^T. \end{aligned}$$

5. A non-linear function of three variables:

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{1 + (x_1 - x_2)^2} + \sin\left(\frac{1}{2}\pi x_2 x_3\right) + \exp\left(-\left(\frac{x_1 + x_3}{x_2} - 2\right)^2\right); \\ \mathbf{x}^0 &= [0, 1, 2]^T; \ \ \mathbf{x}^* \ = \ [1, 1, 1]^T. \end{aligned}$$

6. Freudenstein and Roth function:

$$\begin{aligned} f(\mathbf{x}) &= (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 \\ &\quad + (-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2)^2; \\ \mathbf{x}^0 &= [0.5, -2]^T; \ \ \mathbf{x}^* \ = \ [5, 4]^T; \ \mathbf{x}^*_{\text{local}} = [11.41\ldots, -0.8968\ldots]^T. \end{aligned}$$

7. Powell's badly scaled function:

$$\begin{aligned} f(\mathbf{x}) &= (10\,000 x_1 x_2 - 1)^2 + (\exp(-x_1) + \exp(-x_2) - 1.0001)^2; \\ \mathbf{x}^0 &= [0, 1]^T; \ \ \mathbf{x}^* \ = \ [1.098\cdots \times 10^{-5}, 9.106\ldots]^T. \end{aligned}$$

8. Brown's badly scaled function:

$$\begin{aligned} f(\mathbf{x}) &= (x_1 - 10^6)^2 + (x_2 - 2 \times 10^{-6})^2 + (x_1 x_2 - 2)^2; \\ \mathbf{x}^0 &= [1, 1]^T; \ \ \mathbf{x}^* \ = \ [10^6, 2 \times 10^6]^T. \end{aligned}$$

9. Beale's function:

$$\begin{aligned} f(\mathbf{x}) &= (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 \\ &\quad +(2.625 - x_1(1 - x_2^3))^2; \\ \mathbf{x}^0 &= [1,1]^T; \;\; \mathbf{x}^* \;=\; [3,0.5]^T. \end{aligned}$$

10. Wood's function:

$$\begin{aligned} f(\mathbf{x}) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2 \\ \mathbf{x}^0 &= [-3,-1,-3,-1]^T; \;\; \mathbf{x}^* \;=\; [1,1,1,1]^T. \end{aligned}$$