Optimization exercises

Zsófia Lendek

Preface

This collection of exercises is used to familiarize students with the numerical methods taught in the framework of the Optimization course at the Department of Automation, Technical University of Cluj-Napoca.

As usual, Optimization is quite involved from a mathematical point of view, therefore, in many cases, it is required that the algorithms are implemented on a computer, using Matlab, Mathematica or a similar environment. Here, references are made to Matlab. The exercises go in parallel with the course, from function approximation, through analytical methods, to numerical methods.

Each chapter is structured as follows: an Introduction to the method, an Example, and finally the Exercises. The exercises in a given chapter are quite similar to each other and are solved using the same method.

I hope that this collection will help you better use optimization methods.

Lendek Zsófia
Cluj, 2013

Contents

1 Function approximation **1**
- 1.1 Introduction . . . 1
- 1.2 Example . . . 2
- 1.3 Exercises . . . 4

2 Analytical methods **7**
- 2.1 Introduction . . . 7
- 2.2 Example . . . 8
- 2.3 Exercises: stationary points . . . 11
- 2.4 Exercises: Lagrange multipliers . . . 14

3 Optimization of single variable functions: elimination methods **17**
- 3.1 Introduction . . . 17
- 3.2 Example . . . 18
- 3.3 Exercises . . . 22

4 Newton and gradient methods **25**
- 4.1 Introduction . . . 25
- 4.2 Example . . . 29
- 4.3 Exercises . . . 31

5 Derivative-free methods **35**
- 5.1 Introduction . . . 35
- 5.2 Example . . . 37
- 5.3 Exercises . . . 39

6 Linear programming – the simplex method **45**
- 6.1 Introduction . . . 45
- 6.2 Examples . . . 47
- 6.3 Exercises . . . 50

7 Quadratic programming – the active set method **61**
7.1 Introduction 61
7.2 Example 62
7.3 Exercises 65

Appendices **75**

A Introduction to Matlab **75**
A.1 Introduction 75
A.2 Statements and variables 75
A.3 Entering vectors and matrices 76
A.4 Matlab functions 78
A.5 Polynomials 79
A.6 Loops and logical statements 80
A.7 Plotting 80
A.8 Toolboxes and m-files 82
A.9 Symbolic math 84

Bibliography **87**

Glossary **89**

Chapter 1

Function approximation

1.1 Introduction

Probably the most frequently encountered optimization problem – even though one might not think about it like optimization – is the approximation of a function. Problems that are in this category range from system identification to parameter estimation and can be found in almost every field: control engineering (Khalil, 2002; Narendra and Annaswamy, 1989), mechanical engineering, (Rao et al., 2011; Beck and de Santana Gomes, 2012), civil engineering (Perez and Behdinan, 2007; Kitayama et al., 2011), medicine (Amarantini et al., 2010), geology (Afshari et al., 2011), etc.

Depending on the exact problem formulation and the known part of the problem, e.g., the context of the problem, the structure of the function to be approximated, available data, etc., different methods can be applied. For instance, in system identification frequently used methods are the least squares or recursive least squares, Box-Jenkins (Eykhoff, 1974; Rao, 1978), etc. If a parameter of a dynamic system needs to be determined, observers such as Kalman filters, Luenberger observers, Particle filters (Kalman, 1960; Luenberger, 1966; Arulampalam et al., 2002; Ristic et al., 2004) may represent a solution.

Here, we consider the problem of determining the parameters of a function with a known structure from measured data. Let us consider a discrete-time time-varying function $f(\boldsymbol{x}, k)$, where k denotes the time and $\boldsymbol{x} = [x_1,\, x_2,\, \cdots,\, x_n]^T$ is the vector of unknown parameters, n being the number of parameters. A number m of measurements – usually not precise – of the value of the function $f(\boldsymbol{x}, k_i)$ are available for different time indices $i = 1,\, 2,\, \ldots,\, m$. Our objective is to determine the parameters $\widehat{\boldsymbol{x}}$ such that the function values best approximate the measurements, that is, $f(\widehat{\boldsymbol{x}}, k) \approx f(\boldsymbol{x}, k)$.

First of all, the parameter estimation problem has to be formulated as an optimization problem. This can be done by defining an error function that has to be

minimized. Several error functions can be used, among which the squared error:

$$e_{se}(\boldsymbol{x}, \widehat{\boldsymbol{x}}) = \sum_{i=1}^{m} (f(\boldsymbol{x}, k_i) - f(\widehat{\boldsymbol{x}}, k_i))^2 \tag{1.1}$$

mean squared error

$$e_{mse}(\boldsymbol{x}, \widehat{\boldsymbol{x}}) = \frac{1}{m} \sum_{i=1}^{m} (f(\boldsymbol{x}, k_i) - f(\widehat{\boldsymbol{x}}, k_i))^2 \tag{1.2}$$

absolute error

$$e_{ae}(\boldsymbol{x}, \widehat{\boldsymbol{x}}) = \frac{1}{m} \sum_{i=1}^{m} |f(\boldsymbol{x}, k_i) - f(\widehat{\boldsymbol{x}}, k_i)| \tag{1.3}$$

Depending on the properties of the function, different methods can be used to minimize the error function. Some of these methods will be discussed in later chapters. Here, we will use available already implemented methods to minimize an error function and consequently determine the parameters.

1.2 Example

Consider the function

$$f(\boldsymbol{x}, k) = x_1 k^{x_2} + 3k^2 \tag{1.4}$$

where x_1 and x_2 are unknown parameters, and the measured values of f are shown in Figure 1.1.

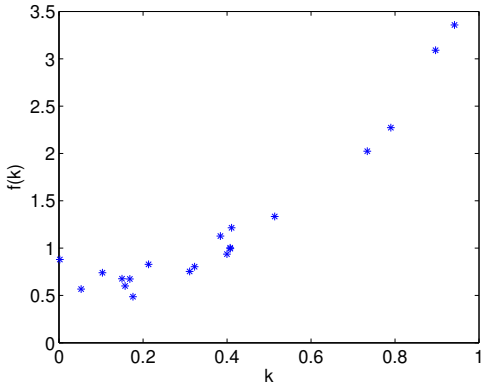


Figure 1.1: Measured values of the function f.

Our objective is to determine the unknown parameters x_1 and x_2. To define the optimization problem, we use e.g., the squared error, i.e., the function to be minimized is

$$e_{se}(\boldsymbol{x}, \widehat{\boldsymbol{x}}) = \sum_{i=1}^{m} (f(\boldsymbol{x}, k_i) - f(\widehat{\boldsymbol{x}}, k_i))^2$$

Most mathematics-oriented programming languages have several optimization methods implemented. We use Matlab's *fminunc* and *fminsearch* functions. *fminunc* is a trust-region/line-search based method, while *fminsearch* uses the Nelder-Mead method. These methods will be presented later on. They are local optimization methods and require defining the objective function and an initial point around which to search for a solution.

Starting from the initial parameter values $\widehat{\boldsymbol{x}}_0 = [0\,0]^T$, both Matlab functions obtain the parameter values $\widehat{\boldsymbol{x}}_0 = [0.5118\ -0.1172]^T$. The squared error is $e = 0.1988$. The comparison of the function values is presented in Figure 1.2.

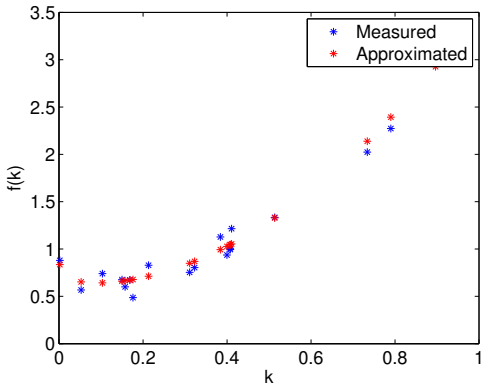


Figure 1.2: Measured and approximated values of the function f.

In many cases, the solution that is obtained is not unique. For instance, consider the function

$$f(\boldsymbol{x}, k) = \sin(x_1 + k) + \cos(x_2 + k) \tag{1.5}$$

with the measured values shown in Figure 1.3.

For the initial condition $[0\,0]^T$ the obtained parameters are $\boldsymbol{x} = [0.7272\,0.7788]^T$, for $[3\,3]^T$ we obtain $[2.3496\,5.4396]$, etc. However, the squared error in all cases is $e = 0.0365$, so these solutions are equivalent. The comparison of the function values is shown in Figure 1.4.

In general, depending on the initial condition and the method used several solutions can be obtained.

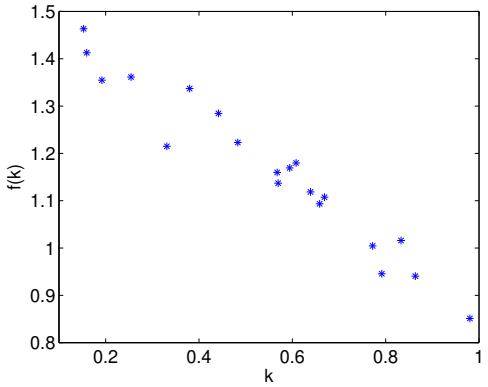


Figure 1.3: Measured values of the function $\sin(x_1 + k) + \cos(x_2 + k)$.

1.3 Exercises

Consider the following functions:

1. $f(\boldsymbol{x}, k) = x_1 k^{x_3} e^{-x_4 k} + x_2$
2. $f(\boldsymbol{x}, k) = \frac{x_1}{1+e^{-\frac{k-x_2}{x_3}}} + x_4$
3. $f(\boldsymbol{x}, k) = x_1 e^{-x_3 k} + x_2$

where $\boldsymbol{x}$ are the unknown parameters, and the measured data from the electronic appendix. The first index in the name of each data file indicates which function should be considered. For instance, *trace1_25* contains the 25th data set for the first function. Determine the parameters of the functions.

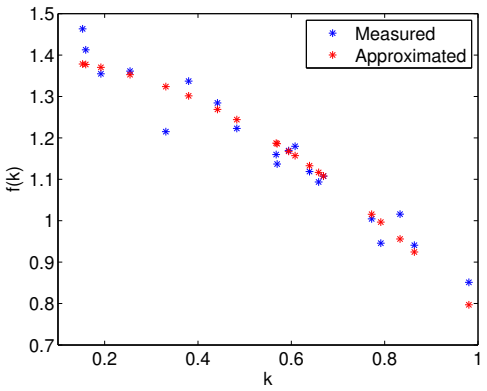


Figure 1.4: Measured and approximated values of the function $\sin(x_1+k)+\cos(x_2+k)$.

Chapter 2

Analytical methods

2.1 Introduction

Analytical methods are classical optimization methods that are used in general for the optimization of continuous and differentiable functions. Consider the twice differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

Stationary points of this function are those points $\boldsymbol{x}_s$ for which the first derivative

$$\frac{\partial f}{\partial \boldsymbol{x}}|_{\boldsymbol{x}_s} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} |_{\boldsymbol{x}_s} = 0$$

For a point $\boldsymbol{x}^*$ to be a local *extremum* (maximum or minimum) of this function, it is *necessary* that $\boldsymbol{x}^*$ is a stationary point. A *sufficient* condition (Rao, 1978) for $\boldsymbol{x}^*$ to be a *local minimum (maximum)* is that the second order derivative (the Hessian)

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

evaluated in $\boldsymbol{x}^*$ is positive (negative) definite. If the Hessian is indefinite (it has both positive and negative eigenvalues) in $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ is a saddle point. If the Hessian is positive or negative semidefinite, no conclusion can be drawn and higher order tests have to be used.

For functions that are not differentiable in every point, an extremum might be located in a point where the function is not differentiable.

In many cases, the objective function has to be optimized in the presence of constraints (Rao, 1978; Raica, 2009). Depending on the constraints, one can use direct substitution, Lagrange multipliers, or, in case of inequality constraints, the Karush-Kuhn-Tucker conditions. Of these, here we consider Lagrange multipliers for *equality constraints*.

Consider the twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, with the equality constraints $g_i(\boldsymbol{x}) = 0$, $i = 1, 2, \ldots, m$, corresponding to the optimization problem

$$\begin{aligned} &\min(\max) f(\boldsymbol{x}) \\ &\text{subject to} \\ &\quad g_i(\boldsymbol{x}) = 0, \quad i = 1, 2, \ldots, m \end{aligned}$$

The *Lagrangian* associated to this optimization problem is

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) \\ &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \end{aligned}$$

Similarly to unconstrained optimization, the *necessary* condition for a point $\boldsymbol{x}_s$ to be the extremum of the function f in the presence of equality constraints is that the first-order derivative of L wrt. $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, evaluated in this point is 0, i.e.,

$$\frac{\partial L}{\partial [\boldsymbol{x}^T \boldsymbol{\lambda}^T]^T}|_{\boldsymbol{x}_s} = 0$$

The *sufficient* condition (Hancock, 1960) for a point $\boldsymbol{x}^*$ to be a local minimum (maximum) is that the roots z_i of the determinant equation

$$\det \left(\begin{array}{cc} \frac{\partial^2 L}{\partial \boldsymbol{x}^2} - zI & \frac{\partial g}{\partial \boldsymbol{x}} \\ \frac{\partial g}{\partial \boldsymbol{x}}^T & 0 \end{array} \right) \Bigg|_{\boldsymbol{x}^*} = 0$$

are all positive (negative). If there are both positive and negative roots, the point is not an extremum.

2.2 Example

Consider the twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $f(\boldsymbol{x}) = (x_1 - 1)^2 + (x_2 - 2)^2$. This function has a minimum in $\boldsymbol{x}^* = [1\, 2]^T$, as can be seen in Figure 2.1.

To prove that this is indeed the minimum of the function, let us calculate the derivatives:

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= 2(x_1 - 1) \\ \frac{\partial f}{\partial x_2} &= 2(x_2 - 2) \end{aligned}$$

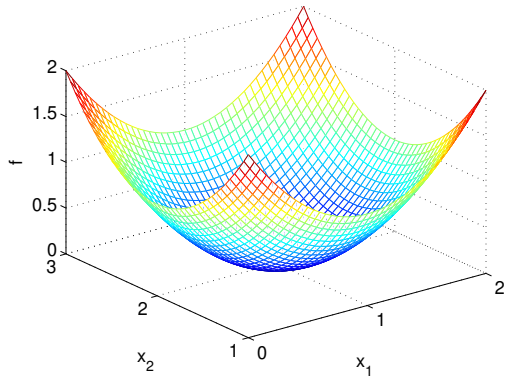


Figure 2.1: The function $(x_1 - 1)^2 + (x_2 - 2)^2$.

From $\frac{\partial f}{\partial \boldsymbol{x}} = 0$ we obtain $x_1 = 1$, $x_2 = 2$. The Hessian is

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

which is positive definite (since it has the eigenvalues 2 and 2) and thus the point $\boldsymbol{x}^* = [1\,2]^T$ is a local minimum.

Consider now the function $f(\boldsymbol{x}) = x_1^2 - x_2^2$. The function is illustrated in Figure 2.2. As can be seen, the point $[0\,0]^T$ is a saddle point.

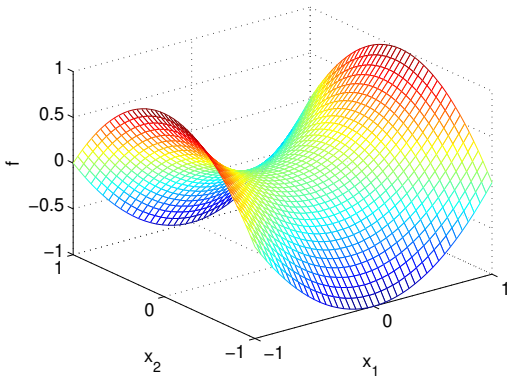


Figure 2.2: The function $x_1^2 - x_2^2$.

The derivatives of the function are

$$\frac{\partial f}{\partial x_1} = 2x_1$$
$$\frac{\partial f}{\partial x_2} = -2x_2$$

and we obtain $x_1 = 0$, $x_2 = 0$. The Hessian is

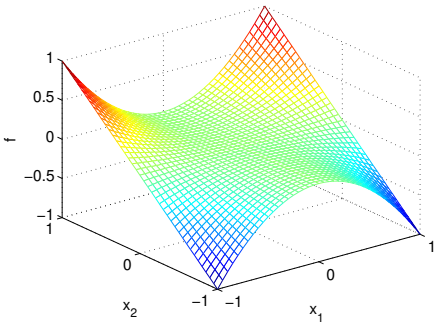
$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

which is indefinite (has a positive and a negative eigenvalue) and thus the point $\boldsymbol{x}^* = [0\,0]^T$ is a saddle point.

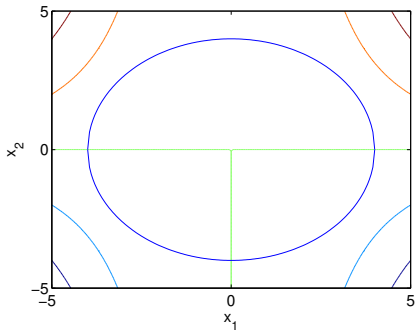
Consider now the following constrained optimization problem:

Maximize $f(\boldsymbol{x}) = x_1^2 x_2$
subject to $x_1^2 + x_2^2 = 1$

The function is represented in Figure 2.2, while the contour plot of the function, together with the constraint is illustrated in Figure 2.2. The Lagrangian is written as



(a) The function $x_1^2 x_2$.



(b) Contour plot of the function $x_1^2 x_2$ and the constraint $x_1^2 + x_2^2 = 16$.

$$L(\boldsymbol{x}, \lambda) = x_1^2 x_2 + \lambda(x_1^2 + x_2^2 - 1)$$

The first derivatives are

$$\begin{aligned} \frac{\partial L}{\partial x_1} &= 2x_1 x_2 + 2\lambda x_1 = 0 \\ \frac{\partial L}{\partial x_2} &= x_1^2 + 2\lambda x_2 = 0 \\ \frac{\partial L}{\partial \lambda} &= x_1^2 + x_2^2 - 1 = 0 \end{aligned} \tag{2.1}$$

while the second derivatives and the derivatives of the constraint are

$$\begin{aligned}
\frac{\partial^2 L}{\partial x_1^2} &= 2x_2 + 2\lambda \\
\frac{\partial^2 L}{\partial x_1 \partial x_2} &= 2x_1 \\
\frac{\partial^2 L}{\partial x_2^2} &= 2\lambda \\
\frac{\partial g}{\partial x_1} &= 2x_1 \\
\frac{\partial g}{\partial x_2} &= 2x_2
\end{aligned} \tag{2.2}$$

To determine the maximum of the function subject to the constraint we have to evaluate for each solution $\boldsymbol{x}_i$ of (2.1) the solution of the determinant equation

$$\det \begin{pmatrix} 2x_{2i} + 2\lambda_i - z_i & 2x_{1i} & 2x_{1i} \\ 2x_{1i} & 2\lambda_i - z_i & 2x_{2i} \\ 2x_{1i} & 2x_{2i} & 0 \end{pmatrix} = 0 \tag{2.3}$$

The solutions of (2.1), the corresponding z and the types of the points are given in Table 2.1. As can be seen, there are in total 6 extrema, out of which 3 are local maxima and 3 are local minima. The maximum value of f is 0.38 and can be obtained either for $x_1 = \frac{\sqrt{6}}{3}$ and $x_2 = \frac{\sqrt{3}}{3}$ or for $x_1 = -\frac{\sqrt{6}}{3}$ and $x_2 = \frac{\sqrt{3}}{3}$.

Table 2.1: Extreme points and their type.

x_1	$x_2 o$	λ	z	Type	$f(\boldsymbol{x})$
0	1	0	2	min	0
0	-1	0	-2	max	0
$\frac{\sqrt{6}}{3}$	$\frac{\sqrt{3}}{3}$	$-\frac{\sqrt{3}}{3}$	-2.3	max	0.38
$\frac{\sqrt{6}}{3}$	$-\frac{\sqrt{3}}{3}$	$\frac{\sqrt{3}}{3}$	2.3	min	-0.38
$-\frac{\sqrt{6}}{3}$	$\frac{\sqrt{3}}{3}$	$-\frac{\sqrt{3}}{3}$	-2.3	max	0.38
$-\frac{\sqrt{6}}{3}$	$-\frac{\sqrt{3}}{3}$	$\frac{\sqrt{3}}{3}$	2.3	min	-0.38

2.3 Exercises: stationary points

This exercise consists in determining the stationary points and their type of a given function.

Consider the following functions $f : \mathbb{R}^n \to \mathbb{R}$:

1. $f(x_1, x_2) = x_1^2 + 2x_1x_2 + x_2^2$
2. $f(x_1, x_2) = 3x_1^2 + 2x_1 - x_2 + x_2^2 + 1$
3. $f(x_1, x_2) = (x_1 - x_2)^2 + (2x_1 + x_2)^2$
4. $f(x_1, x_2) = e^{x_1^2} + x_2^2 - 3 + 2x_2$
5. $f(x_1, x_2) = 2x_1^2 + 3x_1x_2 + x_2 + x_2^2 - 1$
6. $f(x_1, x_2) = (x_1^2 - 1)^2 + x_2^2 - 3x_2 + 1$
7. $f(x_1, x_2) = x_1^2e^{x_1} + 51x_2 + x_2^4 + 3$
8. $f(x_1, x_2) = x_1^4 + 2x_1^2x_2 + x_2^2 + 3$
9. $f(x_1, x_2) = e^{x_1^2-3} + x_2^2 - 3x_2$
10. $f(x_1, x_2) = x_1x_2^3 + 2x_1^2 + 2x_2^4 - 5$
11. $f(x_1, x_2) = x_1^2x_2 + 6x_2^2 - 3x_1x_2 + 4x_1^4 + 3x_1^2$
12. $f(x_1, x_2, x_3) = 2x_1^2 + 3x_2^2 + x_2x_3 + 6x_3^2$
13. $f(x_1, x_2) = x_1^6 + 3x_1x_2 + x_2^2$
14. $f(x_1, x_2) = 18x_1^2 + 20x_2 + x_1x_2 + x_2^2$
15. $f(x_1, x_2) = 5x_1^4 + x_1^2x_2 + x_2^2 - 3x_2 + 1$
16. $f(x_1, x_2) = (\cos(2\pi))^{x_1} + x_1x_2 + x_2^2 + x_1^2$
17. $f(x_1, x_2) = 32x_1x_2^2 + 9x_2^2 + 18x_1^2 + 3$
18. $f(x_1, x_2) = x_1^3 + 3x_1^4 + 31x_1x_2 + x_2^2$
19. $f(x_1, x_2) = 6x_1^2x_2^2 + 3x_1x_2 - 1$
20. $f(x_1, x_2) = 4x_1^3 + 6x_1^4 + 3x_1x_2^2 + x_2^4$
21. $f(x_1, x_2) = x_1^4 + 2x_1^2x_2 + x_2^6 + 6x_1x_2^2 + 3$
22. $f(x_1, x_2) = x_1x_2 + x_2^2 + x_2^4 + 3x_1^2 - 1$
23. $f(x_1, x_2) = 2x_1^2x_2 + 31x_1^4 + 18x_2^2 + 3$
24. $f(x_1, x_2) = 11x_1 + 22x_1^2x_2 + x_2^2 + 31x_1^2$
25. $f(x_1, x_2) = x_1^2x_2 + 5x_2^3 + x_2^4 + 3x_1^2$

26. $f(x_1, x_2) = x_1x_2 + 3x_2^2 + 4x_2^4 + x_1^2$
27. $f(x_1, x_2) = x_1^2 + x_2 + 3x_1 + 6$
28. $f(x_1, x_2) = 64x_1^2 + 64x_2^2 + 128x_1x_2 - 16$
29. $f(x_1, x_2) = 10x_1^2 + 6x_2^2 + 8x_1^4x_2^4 + 24$
30. $f(x_1, x_2) = 81x_1^2 + 27x_1x_2 + 18x_2^2 + x_2^4 - 9$
31. $f(x_1, x_2) = x_1x_2^3 + 9x_1^2 - 3x_2^2 + 8$
32. $f(x_1, x_2) = x_1^2 - x_2^2 + 8x_1x_2 - x_2^4 + 1$
33. $f(x_1, x_2) = -x_1^2 - x_2^2 + 18x_1x_2 - 3$
34. $f(x_1, x_2) = 1024x_1 - 512x_1x_2 + 2x_2^2 + 2x_1$
35. $f(x_1, x_2) = 5x_1^2 + 3x_1x_2 - x_2^2 + x_2^4$
36. $f(x_1, x_2) = x_1^3 + 6x_1^4 - 3x_2^2 + 2x_2^6$
37. $f(x_1, x_2, x_3) = x_2^2 + x_3^2 + 3x_1x_2 - x_3$
38. $f(x_1, x_2) = (x_1 + 43x_2)^3 + 3x_1^2 - 5$
39. $f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 - (x_1 + x_2)^2$
40. $f(x_1, x_2, x_3) = 33x_1^2 + (x_2 - x_3)^2 + x_1x_2$
41. $f(x_1, x_2, x_3) = x_1x_3 + 3x_2 + x_3^2 + x_1^2 + x_2^2$
42. $f(x_1, x_2, x_3) = x_1x_2x_3 + x_1^2x_2^2 + 5x_3^2 + 1$
43. $f(x_2, x_4) = x_2^2 + x_4^3 + x_2^6x_4^2$
44. $f(x_1, x_2) = e^{x_1^2} + 3x_1^2 + 1$
45. $f(x_1, x_2) = 2x_2^2 + 3x_2^4 + 5x_1^2 + 3$
46. $f(x_1, x_2) = -x_1^3 + x_1^4 + x_2 + x_2^2$
47. $f(x_1, x_2) = x_1^2x_2^2 + 113x_1x_2 - 1$
48. $f(x_1, x_2) = x_1^3 + 6x_1^4 + x_2^2 + x_2^4$
49. $f(x_1, x_2) = x_1^4 + 2 + x_2^6 + x_1x_2 - 10$
50. $f(x_1, x_2) = -10x_2 + x_2^2 + x_2^4 + 3x_1^2 - 1$
51. $f(x_1, x_2) = x_1^2x_2 + 11x_1^4 + 18x_2^2 - 15$
52. $f(x_1, x_2) = x_1 + 2x_1^2x_2 + 2x_2^2 + 4x_1^2$

2.4 Exercises: Lagrange multipliers

Consider now the functions from Section 2.4 and the following equality constraints. Determine the minima or maxima of the functions subject to the constraint(s).

1. $x_1 + x_2 = 1$
2. $x_1^2 + x_2 = 1$
3. $2x_1 - 3x_2 = 2$
4. $x_1 = 5x_2$
5. $e_1^x = 1$
6. $x_1 + 2x_2 = 3$
7. $x_1 + 3x_2 = 2$
8. $x_1 - 2x_2 = 5$
9. $x_1 + x_2^2 = 3$
10. $2x_1 - 3x_2 = 5$
11. $x_1 - 2x_2 = 0$
12. $2x_1 - x_1^2 = -5$
13. $x_1^2 + x_2^2 = 0$
14. $x_1x_2 + 3x_2^2 + 1 = 0$
15. $x_1 + 5x_2 = 2$
16. $5x_1 + 3x_2 = 10$
17. $6x_1 + 4x_2 = 12$
18. $21x_1 - 3x_2 = 5$
19. $x_1 + 10x_2 = 1$
20. $x_1 + 8x_2 = 0$
21. $3x_1 + 5x_2 = 3$
22. $8x_1 + 15x_2 - x_3 = 0$

23. $x_1^2 = x_2$
24. $x_2^2 = x_1$
25. $x_1 - x_2 + 1 = 0$
26. $x_1 x_2 + x_2 = 0$
27. $x1 + 15x_2 = 3$
28. $x_1^2 + x_2^2 = 1$
29. $(x_1 - 1)^2 + (x_2 - 1)^2 = 1$
30. $64x_1^2 + 16x_1 + 1 = 0$
31. $4x_1 + 3x_2 = 1$
32. $x_1 + 6x_2 = 15$
33. $x_1 + 6x_2 = 10$
34. $15x_1 - x_2 = 1$
35. $6x_1 - 7x_2 = 8$
36. $5x_1 - 25x_2 = 1$
37. $4x_1 + 7x_2 = 11$
38. $x_1^2 + 2x_2^2 = 3$
39. $11x_1 + 19x_2 = 23$
40. $3x_1 + 9x_2 = 13$
41. $3x_1 + 5x_2 = 1$
42. $50x_1 + 125x_2 = 25$
43. $x_1 + 10x_2 = 11$
44. $7x_1 - x_2 = 3$
45. $8x_1 - 4x_2 = 1$
46. $x_1 + 6x_2 = -5$
47. $25x_1 - x_2 + 3 = 0$

48. $x_1 + 16x_2 = 1$
49. $x_1 + 3x_2 = 100$
50. $31x_1 + 19x_2 = 2$
51. $2x_1 + 8x_2 = 16x_3$
52. $3x_1 + x_2 + x_3 = 0$
53. $118x_1 + 59x_2 = 236$
54. $x_1 + 5x_2 - x_3 = 1$
55. $x_1 + x_2 + x_3 + x_4 = 5$
56. $x_2 + x_3 = -x_1$
57. $x_1 - 2x_2 = 3x_3$
58. $x_1 - x_2 + 3x_3 = 6$
59. $10x_1 + 5x_2 + x_3 = 1$
60. $3x_2 + 5x_3 = 0$

Chapter 3

Optimization of single variable functions: elimination methods

3.1 Introduction

Although analytical methods are able to provide an exact minimum or maximum, their use can be cumbersome, in particular for nonlinear functions and constraints. This is why numerical methods are widely used. In this chapter, we consider the simplest case of numerical optimization, when the function to be optimized depends on a single variable and is unimodal, i.e., it has a unique extremum on a given interval. Our goal is to find this unique extremum.

Consider the function $f : [a,\, b] \to \mathbb{R}$. Without loss of generality, let us assume that the function f has a unique minimum on $[a,\, b]$ (see e.g., Figure 3.1). The following methods find this minimum with a tolerance ε by *eliminating* parts of the interval in several steps.

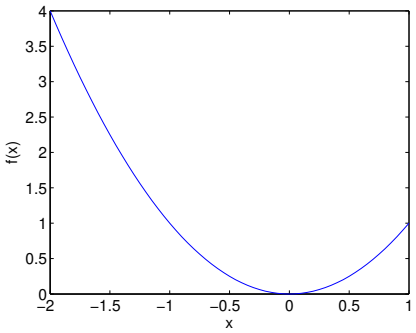


Figure 3.1: A function with a unique minimum on the interval $[-2,\, 1]$.

For the elimination, one has to define two, partly overlapping intervals. The func-

tion is evaluated in the endpoints of the intervals. Since there is a unique minimum, one of the intervals contains a point with smaller function value. This interval is retained, while the remaining part is eliminated.

For instance, for the function in Figure 3.1, if one chooses the points $x_1 = -1.5$ and $x_2 = 0.5$, corresponding to the intervals $[-2,\ 0.5]$ and $[-1.5,\ 1]$ the values of the function in these points are $f(x_1) = 2.25$ and $f(x_2) = 0.25$. Since $f(x_2) < f(x_1)$, the interval $[-1.5,\ 1]$ is retained, and the part $[-2,\ -1.5]$ is eliminated.

Two methods for choosing the intervals (or points) are summarized in the Algorithms 3.1 and 3.2. Algorithm 3.1 is based on the "golden ratio", while Algorithm 3.2 uses the ratio of consecutive Fibonacci numbers to determine the points where the function should be evaluated.

Algorithm 3.1 Golden section

Input: Objective function $f(x)$, boundaries a and b, and tolerance ε

```
d = b − a
while b − a ≥ ε do
    d ← 0.618 × d
    x1 ← b − d
    x2 ← a + d
    if f(x1) ≤ f(x2) then
        b ← x2
    else
        a ← x1
    end if
end while
```

Output: Reduced interval $[a, b]$

Remarks: Elimination methods cannot be used if there are several minima or maxima. These methods do not find an exact value of the extremum, only an interval in which the extremum lies.

3.2 Example

Consider the function $f : [-2,\ 1] \to \mathbb{R}$, $f(x) = x^2$, represented in Figure 3.1. As can be seen, this function has a unique minimum on the interval $[-2,\ 1]$. Our goal is to find this minimum with a tolerance $\varepsilon = 0.3$, using Algorithm 3.1. Algorithm 3.2 can be similarly employed.

1. Inputs: $f(x) = x^2$, $a = -2$, $b = 1$, $\varepsilon = 0.3$. It follows that $d = 1 - (-2) = 3$

Algorithm 3.2 Fibonacci search

Input: Objective function $f(x)$, boundaries a and b, and tolerance ε

$F_1 = 2 \;\; F_2 = 3$
$n = 2$
while $b - a \geq \varepsilon$ **do**
  $d = b - a$
  $x_1 \leftarrow b - d\frac{F_{n-1}}{F_n}$
  $x_2 \leftarrow a + d\frac{F_{n-1}}{F_n}$
  if $f(x_1) \leq f(x_2)$ **then**
    $b \leftarrow x_2$
  else
    $a \leftarrow x_1$
  end if
  $n = n + 1$
  $F_n = F_{n-1} + F_{n-2}$
end while

Output: Reduced interval $[a, b]$

2. Step 1:

$$
\begin{aligned}
d &= (1 - (-2))0.618 = 1.8540 \\
x_1 &= 1 - 1.8540 = -0.8540 \\
x_2 &= -2 + 1.8540 = -0.1460 \\
f(x_1) &= 0.7293 \\
f(x_2) &= 0.0213
\end{aligned}
$$

   Since $f(x_2) < f(x_1)$, the retained interval is $[-0.8540,\ 1]$ (see Figure 3.2) and a and b are modified accordingly: $a = -0.8540$, $b = 1$. The remaining part has been eliminated.

3. Step 2:

$$
\begin{aligned}
d &= 0.618d = 1.1458 \\
x_1 &= b - d = -0.1458 \\
x_2 &= a + d = 0.2918 \\
f(x_1) &= 0.0212 \\
f(x_2) &= 0.0851
\end{aligned}
$$

   Since $f(x_1) < f(x_2)$, the retained interval is $[-0.8540,\ 0.2918]$ (see Figure 3.3) and a and b are modified accordingly: $a = -0.8540$, $b = 0.2958$. The remaining part has been eliminated.

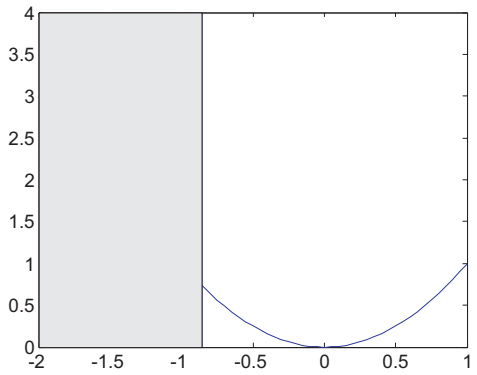


Figure 3.2: After Step 1.

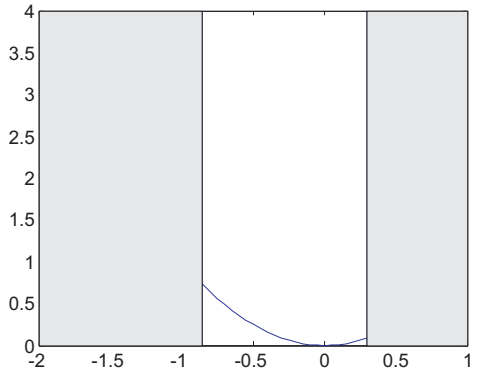


Figure 3.3: After Step 2.

4. Step 3:

$$
\begin{aligned}
d &= 0.618d = 0.7081 \\
x_1 &= b - d = -0.4163 \\
x_2 &= a + d = -0.1459 \\
f(x_1) &= 0.1733 \\
f(x_2) &= 0.0212
\end{aligned}
$$

Since $f(x_2) < f(x_1)$, the retained interval is $[-0.4163, 0.2918]$ (see Figure 3.4) and a and b are modified accordingly: $a = -0.4163$, $b = 0.2958$. The remaining part has been eliminated.

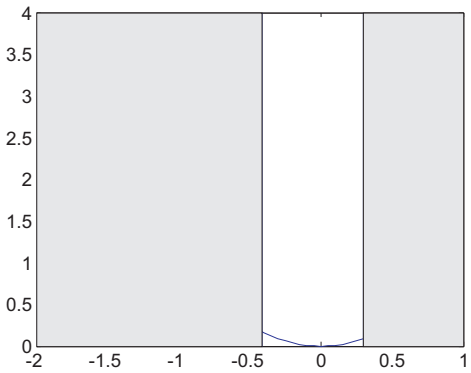


Figure 3.4: After Step 3.

5. Step 4:

$$
\begin{aligned}
d &= 0.618d = 0.4376 \\
x_1 &= b - d = -0.1458 \\
x_2 &= a + d = 0.0213 \\
f(x_1) &= 0.0213 \\
f(x_2) &= 4.5 \cdot 10^{-4}
\end{aligned}
$$

Since $f(x_2) < f(x_1)$, the retained interval is $[-0.1458, 0.2918]$ (see Figure 3.5) and a and b are modified accordingly: $a = -0.1458$, $b = 0.2958$. The remaining part has been eliminated.

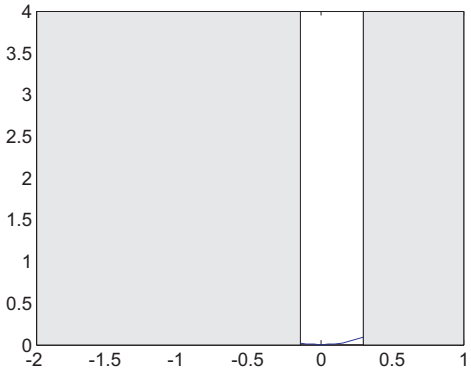


Figure 3.5: After Step 4.

6. Step 5:

$$d = 0.618d = 0.2704$$
$$x_1 = b - d = 0.0213$$
$$x_2 = a + d = 0.1246$$
$$f(x_1) = 4.5 \cdot 10^{-4}$$
$$f(x_2) = 0.0155$$

Since $f(x_1) < f(x_2)$, the retained interval is $[-0.1458, 0.1246]$ (see Figure 3.6) and a and b are modified accordingly: $a = -0.1458$, $b = 0.1246$. The remaining part has been eliminated. The length of the remaining interval is less than the tolerance $\varepsilon = 0.3$, therefore the algorithm stops here. Any point taken from the interval $[-0.1458, 0.1246]$ can be considered the minimum of the function f with tolerance ε.

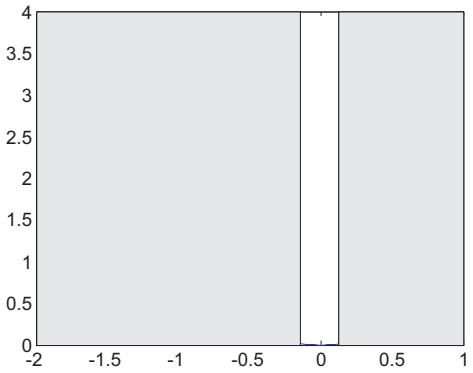


Figure 3.6: After Step 5.

3.3 Exercises

Consider the following single variables functions $f : [a, b] \to \mathbb{R}$. Verify whether they have a unique minimum or a maximum on the given interval. Implement the Fibonacci and the Golden Section method and find the unique extremum on the given interval.

1. $f(x) = x^2 - 2x - 5$, $a = 0$, $b = 2$
2. $f(x) = 3x + x^3 + 5$, $a = -4$, $b = 4$
3. $f(x) = \sin(x) + 3x^2$, $a = -2$, $b = 2$
4. $f(x) = e^{x^2} + 3x$, $a = -1$, $b = 1$

5. $f(x) = x^3 - 3x$, $a = -3$, $b = 0$
6. $f(x) = x^3 - 3x$, $a = 0$, $b = 3$
7. $f(x) = \sin(x)$, $a = 0$, $b = \pi$
8. $f(x) = \sin(2x)$, $a = 0$, $b = 2$
9. $f(x) = \cos(x)$, $a = \pi/2$, $b = 3\pi/2$
10. $f(x) = \tan^2(x)$, $a = -\pi/4$, $b = \pi/4$
11. $f(x) = e^x \sin(x)$, $a = 0$, $b = \pi$
12. $f(x) = x^4 - 3x^2$, $a = -4$, $b = 0$
13. $f(x) = x^4 - 3x^2$, $a = 0$, $b = 4$
14. $f(x) = x^5 - 5x^3$, $a = -4$, $b = 0$
15. $f(x) = x^5 - 5x^3$, $a = 0$, $b = 4$
16. $f(x) = x^6 + 5x^2$, $a = -1$, $b = 1$
17. $f(x) = x^3 - 9x$, $a = -3$, $b = 0$
18. $f(x) = x^3 - 9x$, $a = 0$, $b = 3$
19. $f(x) = x^3 + 9x$, $a = -1$, $b = 1$
20. $f(x) = 3x^4 - 6x^2$, $a = -3$, $b = 0$
21. $f(x) = 3x^4 - 6x^2$, $a = 0$, $b = 3$
22. $f(x) = e^x + e^{-x}$, $a = -2$, $b = 2$
23. $f(x) = e^x - e^{-x}$, $a = -2$, $b = 2$
24. $f(x) = \begin{cases} \frac{\sin(x)}{x}, \text{ if } x \neq 0 \\ 1, \text{ otherwise} \end{cases}$, $a = -\pi/2$, $b = \pi/2$
25. $f(x) = \sin^2(x)$, $a = -\pi/2$, $b = \pi/2$
26. $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $a = -1$, $b = 1$
27. $f(x) = 6x^2 - 12x$, $a = -4$, $b = 0$
28. $f(x) = 6x^2 - 12x$, $a = 0$, $b = 4$

29. $f(x) = \sin(x)(6x^2 - 12x)$, $a = -4$, $b = -1$
30. $f(x) = \sin(x)(6x^2 - 12x)$, $a = 0$, $b = 2$
31. $f(x) = \cos(x)(x^2 - 5)$, $a = -3$, $b = -1$
32. $f(x) = \cos(x)(x^2 - 5)$, $a = -1$, $b = 1$
33. $f(x) = \cos(x) + \sin(x)$, $a = -1$, $b = 3$
34. $f(x) = |x|$, $a = -1$, $b = 1$
35. $f(x) = \text{sign}(x)(x^2 - 3x)$, $a = -4$, $b = 0$
36. $f(x) = |x|(x^2 - 3x)$, $a = 0$, $b = 4$
37. $f(x) = x^4 - 3x^2 - 2$, $a = -2$, $b = 0$
38. $f(x) = (x^6 - 2x^2)\sin(x)$, $a = -2$, $b = 0$
39. $f(x) = (6x^3 - 3x)\cos(x)$, $a = 0.5$, $b = 2$
40. $f(x) = (35x^3 - x)\text{sign}(x)$, $a = -1$, $b = 0$
41. $f(x) = (35x^3 - x)\text{sign}(x)$, $a = 0$, $b = 2$
42. $f(x) = x^4 - 2x^3$, $a = 0$, $b = 2$
43. $f(x) = (6x^7 - 42x)\sin(x)$, $a = 0$, $b = 2$
44. $f(x) = (3x^5 - 16x)\text{sign}(x)$, $a = -1$, $b = 1$
45. $f(x) = x^4 - 3x^2 + 5$, $a = 0$, $b = 2$
46. $f(x) = 6x^8 + 9x^2 - 1$, $a = -2$, $b = 2$
47. $f(x) = 12x^2 - 8x$, $a = -2$, $b = 2$
48. $f(x) = 5x^6 - 3x$, $a = 0$, $b = 2$
49. $f(x) = (5x^6 - 8x)\sin(x)$, $a = 0$, $b = 2$
50. $f(x) = (3x^4 - 12x)\cos(x)$, $a = -2$, $b = 0$
51. $f(x) = (3x^2 + 2x)\text{atan}(x)$, $a = -0.5$, $b = 2$

Chapter 4

Newton and gradient methods

4.1 Introduction

Let us consider now unconstrained optimization of multivariable functions, i.e., optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$$

Newton and gradient methods are widely used for solving this minimization problem when the gradient and/or the Hessian can be computed relatively easy. Newton methods are based on the quadratic approximation of the objective function, while gradient methods rely on determining the best direction and a step-size for finding the minimum, i.e., on a linear approximation.

Since these are iterative methods, it is important to define suitable stopping conditions. Generally used stopping criteria are:

- Variation in successive points: $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \varepsilon$
- Variation in the value of the function: $|f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)| < \varepsilon$
- Gradient: $\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\| < \varepsilon$

where ε represents the desired tolerance. In what follows, unless otherwise specified, the stopping criteria above will be used.

The Newton method is summarized in Algorithm 4.1. As already mentioned, this method relies on a second-order approximation of the objective function, and necessitates the computation of both the gradient and the Hessian of the function in each iteration. When it is computationally expensive to invert (or even compute) the Hessian in each step, but convergence speed is not an issue, the modified Newton method can be used. The modified Newton method uses the inverse of the Hessian in the initial point in every step, i.e., instead of $H^{-1}(\boldsymbol{x}_k)$, $H^{-1}(\boldsymbol{x}_0)$ is used throughout the algorithm (see Algorithm 4.2).

Algorithm 4.1 Newton method

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$, Hessian $H(\boldsymbol{x}) = \frac{\partial^2 f}{\partial \boldsymbol{x}^2}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε
  Set $k = 0$
  while Stopping criterion is not satisfied **do**
    Compute a new point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H^{-1}(\boldsymbol{x}_k)\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$
    Set $k \leftarrow k + 1$
  end while
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

Algorithm 4.2 Modified Newton method

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$, Hessian $H(\boldsymbol{x}) = \frac{\partial^2 f}{\partial \boldsymbol{x}^2}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε
  Set $k = 0$
  while Stopping criterion is not satisfied **do**
    Compute a new point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H^{-1}(\boldsymbol{x}_0)\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$
    Set $k \leftarrow k + 1$
  end while
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

It has to be noted that the Newton method stops in stationary points, not necessarily minima. Once the algorithm stops, the type of the point has to be determined by evaluating the eigenvalues of the Hessian in the point found, as explained in Chapter 2.

A possible problem in the implementation of the Newton method is represented by the Hessian becoming singular, or, in case of the modified Newton method, the initial Hessian having the same problem. A solution for this problem is the Levenberg-Marquardt algorithm (see Algorithm 4.3), which, instead of the Hessian $H(\boldsymbol{x}_k)$, uses $\lambda I + H(\boldsymbol{x}_k)$, where λ is a suitable chosen parameter. If λ is much smaller than the eigenvalues of the Hessian, then the Hessian dominates, well illustrating the nice properties of the Newton method. On the other hand, if λ is much larger than the eigenvalues of the Hessian, the latter can be neglected, leading in effect to a gradient method, as described below.

Gradient methods (see Algorithm 4.4) only use the gradient of the objective function, and search for the extremum of the function along a direction given by this gradient. If one searches for maxima, the direction is the gradient (the steepest ascent method), while if one searches for minima, the direction is the negative of the gradient (the steepest descent method). For both cases, a step size has to be chosen or computed. If a constant step-size is chosen, a possible problem that has to be over-

Algorithm 4.3 Levenberg-Marquardt algorithm

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$, Hessian $H(\boldsymbol{x}) = \frac{\partial^2 f}{\partial \boldsymbol{x}^2}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε, constant λ
  Set $k = 0$
  while Stopping criterion is not satisfied **do**
    Compute a new point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\lambda I + H(\boldsymbol{x}_k))^{-1} \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$
    Set $k \leftarrow k + 1$
  end while
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

come is oscillation around the minimum. When oscillation is detected, the step-size can be reduced, or the algorithm stopped. The step size can also be computed using an elimination method. Once the direction is determined, the next point only depends on the step-size. Thus, in order to find the optimal step-size, one can implement either the golden section or the Fibonacci method and minimize the objective function as a function of the step-size. It must be kept in mind that the step size has to be positive in order to not to change the direction.

Algorithm 4.4 Steepest descent

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε
  Set $k = 0$
  repeat
    Find the step length $s_k > 0$ by minimizing

$$f\left(\boldsymbol{x}_k - s_k \frac{\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)}{\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\|}\right)$$

    Compute new point:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - s_k \frac{\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)}{\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\|}$$

    Set $k \leftarrow k + 1$
  until Stopping criterion is satisfied
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

The convergence of the steepest descent method can be quite slow, if the problem is poorly scaled, but it can be greatly improved if one uses conjugate gradient methods. These use a "conjugate direction" instead of the gradient of the function. The

search direction in each step is given by

$$\boldsymbol{d}_{k+1} = -\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) + \beta_k \boldsymbol{d}_k$$

where $\boldsymbol{d}_k$ is the current direction and β_k is computed in each step using either

- Fletcher-Reeves formula

$$\beta_k = \frac{\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1})\|^2}{\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\|^2} \tag{4.1}$$

- Polak-Ribière formula:

$$\beta_k = \frac{\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1})^T \left(\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) - \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\right)}{\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)^T \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)} \tag{4.2}$$

- Hestenes-Stiefel formula:

$$\beta_k = \frac{\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1})^T \left(\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) - \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\right)}{\boldsymbol{d}_k^T \left(\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) - \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\right)} \tag{4.3}$$

The steps of the conjugate gradient method are given in Algorithm 4.5.

A shortcoming of conjugate gradient methods is that if the step size computed in each iteration is not precise, the errors accumulate. To overcome this, the method may be restarted after every n steps, n being the number of variables. This means that after n steps, the direction is reset to the gradient in the current point.

Even more efficient than conjugate gradient methods are the so-called quasi-Newton methods. These can be regarded as approximations of the Newton method, as they use the information of the gradient to approximate the second order derivative, e.g., by using the Davidon-Fletcher-Powell or Broyden-Fletcher-Goldfarb-Shanno formulas. However, they also compute a step-size as gradient methods do. The general algorithm is described in Algorithm 4.6.

The Davidon-Fletcher-Powell (DFP) formula is given by:

$$B_{k+1} = B_k + \frac{\Delta \boldsymbol{x}_k \Delta \boldsymbol{x}_k^T}{\Delta \boldsymbol{x}_k^T \Delta G_k} - \frac{B_k \Delta G_k (B_k \Delta G_k)^T}{\Delta G_k^T B_k \Delta G_k} \tag{4.4}$$

while the Broyden-Fletcher-Goldfarb-Shanno (BFGS) relation is

$$B_{k+1} \quad = B_k + \frac{\Delta G_k \Delta G_k^T}{\Delta G_k^T \Delta \boldsymbol{x}_k} - \frac{B_k \Delta \boldsymbol{x}_k (B_k \Delta \boldsymbol{x}_k)^T}{\Delta \boldsymbol{x}_k^T B_k \Delta \boldsymbol{x}_k} \tag{4.5}$$

where B_k is the approximation of the inverse of the Hessian in the kth step, $\Delta \boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, and $\Delta G_k = \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) - \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$.

Algorithm 4.5 Conjugate gradient method for minimization

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε
First direction $\boldsymbol{d}_0 = -\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_0)$
repeat
Find the step length $s_k > 0$ by minimizing

$$f(\boldsymbol{x}_k + s_k \boldsymbol{d}_k)$$

Compute

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_k \boldsymbol{d}_k$$

Compute β_k using (4.1), (4.2) or (4.3).
Compute the new direction

$$\boldsymbol{d}_{k+1} = -\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) + \beta_k \boldsymbol{d}_k$$

Set $k \leftarrow k + 1$
until Stopping criterion is satisfied.
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

4.2 Example

Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$, $f(\boldsymbol{x}) = x_1^4 + 2x_1^2 x_2 + 2x_2^2 - x_2 + 3$, represented in Figure 4.1. This function has a local minimum in $\begin{pmatrix} 0 & \frac{1}{4} \end{pmatrix}^T$, verifiable using analytic methods. Here, we attempt to find this minimum using the numerical methods described above, with the stopping criterion being $\|\frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\| < 0.001$. In all cases, we start the search from the initial point $\begin{pmatrix} 2 & 2 \end{pmatrix}^T$.

The gradient of the function is

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{pmatrix} 4x_1^3 + 4x_1 x_2 \\ 2x_1^2 + 4x_2 - 1 \end{pmatrix}$$

and the Hessian is

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \begin{pmatrix} 12x_1^2 + 4x_2 & 4x_1 \\ 4x_1 & 4 \end{pmatrix}$$

The trajectories given by the successive points found by the Newton and modified Newton method are presented in Figure 4.2. As can be seen, both methods converge to the same point, the local minimizer of the function. However, while the Newton method requires only 9 steps, the modified Newton method takes 352 steps. The computation time is 0.01sec for the Newton method and 0.40sec for the modified Newton

Algorithm 4.6 Quasi-Newton method for minimization

Input: Objective function $f(\boldsymbol{x})$, gradient $\frac{\partial f}{\partial \boldsymbol{x}}$
Input: Initial point $\boldsymbol{x}_0$, tolerance ε
Set $k = 0$ and $B_0 = I$
repeat
Compute the search direction:

$$\boldsymbol{d}_k = -B_k \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$$

Find the step length $s_k > 0$ by minimizing $f(\boldsymbol{x}_k + s_k \boldsymbol{d}_k)$
Compute a new point $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_k \boldsymbol{d}_k$
Compute the differences:

$$\Delta \boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$$

$$\Delta G_k = \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_{k+1}) - \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$$

Update B_{k+1} using DFP (4.4) or BFGS (4.5).
Set $k \leftarrow k + 1$
until Stopping criterion is satisfied.
Output: Minimum $\boldsymbol{x}^*$ with tolerance ε

method. This is because, although the Hessian does not have to be evaluated after the first step, the number of iterations increases by almost two orders of magnitude.

Let us now exemplify the steepest descent methods with constant and computed step-size, using un-normalized gradients. The trajectories given by the successive points found by the two methods are presented in Figure 4.3. Both methods converge to the same point. For this specific example, the steepest descent with variable step requires 9 steps, 0.0sec, and with fixed step size $s = 0.05$, 97 steps are made in 0.06sec. For computing the step size, the golden section method has been implemented, and a step-size smaller than 2 has been searched for with tolerance $\varepsilon = 0.001$.

Consider now the conjugate gradient method using the Fletcher-Rieves formula. The trajectory is shown in Figure 4.4. The method required 11 steps and the elapsed time was 0.02sec.

Finally, let us see the quasi-Newton method with the Davidon-Fletcher-Powell formula. The trajectory is shown in Figure 4.5. The method required 6 steps and the elapsed time was 0.01sec.

For this particular example all the methods perform is a similar way, the performance differences being very small. However, in general, different methods may converge to different local minimizers, or even diverge. This also holds if different

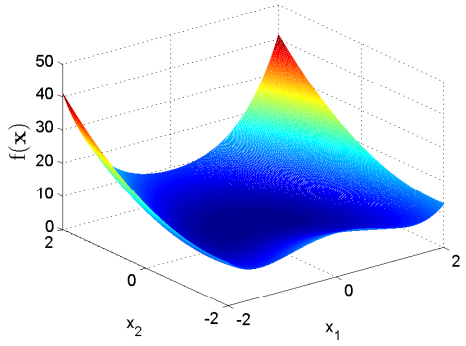


Figure 4.1: Graphical representation of the function $f(\boldsymbol{x}) = x_1^4 + 2x_1^2x_2 + 2x_2^2 - x_2 + 3$.

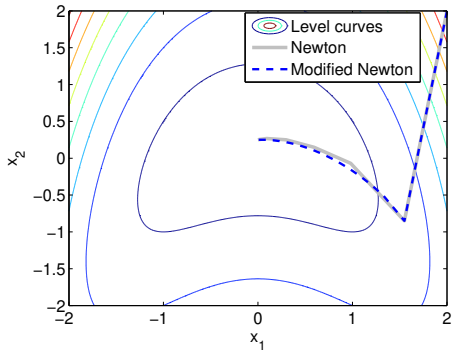


Figure 4.2: Trajectories obtained using the Newton and modified Newton methods.

stopping criteria are used.

4.3 Exercises

Consider the following functions $f : \mathbb{R}^n \to \mathbb{R}$. Implement and compare the Newton, modified Newton, fixed-step gradient, variable-step gradient methods, conjugate gradient and quasi-Newton methods for finding a local minima. Use the golden section or the Fibonacci method to determine the step-size at each step.

1. $f(x_1, x_2) = x_1^6 + 3x_1x_2 + x_2^2$
2. $f(x_1, x_2) = 18x_1^2 + 20x_2^4 + x_1x_2 + x_2^2$
3. $f(x_1, x_2) = 5x_1^4 + x_1^2x_2 + x_2^2 - 3x_2 + 1$

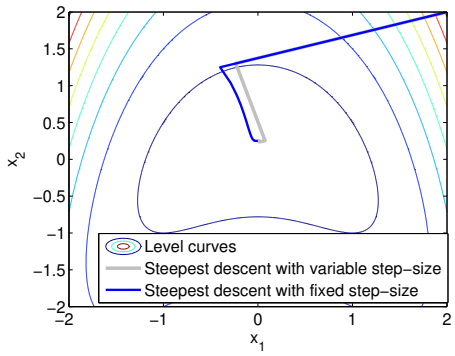


Figure 4.3: Trajectories obtained by the steepest descent method with fixed and variables step-sizes.

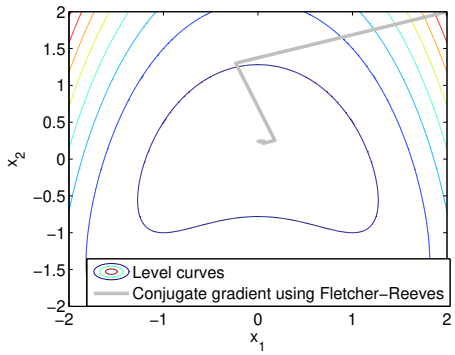


Figure 4.4: Trajectory obtained by a conjugate gradient method.

4. $f(x_1, x_2) = (\cos(2\pi))^{x_1} + x_1 x_2 + x_2^2 + x_1^2$
5. $f(x_1, x_2) = 32x_1x_2^2 + 9x_2^2 + 18x_1^2 + 3$
6. $f(x_1, x_2) = x_1^3 + 3x_1^4 + 31x_1x_2 + x_2^2$
7. $f(x_1, x_2) = 10x_1^2 + 6x_2^2 + 8x_1^4x_2^4 + 24$
8. $f(x_1, x_2) = 81x_1^2 + 27x_1x_2 + 18x_2^2 + x_2^4 - 9$
9. $f(x_1, x_2) = x_1x_2^3 + 9x_1^2 - 3x_2^2 + 8$
10. $f(x_1, x_2) = x_1^2 - x_2^2 + 8x_1x_2 - x_2^4 + 1$
11. $f(x_1, x_2) = -x_1^2 - x_2^2 + 18x_1x_2^3 - 3$
12. $f(x_1, x_2) = 5x_1^2 + 3x_1x_2 - x_2^2 + x_2^4$

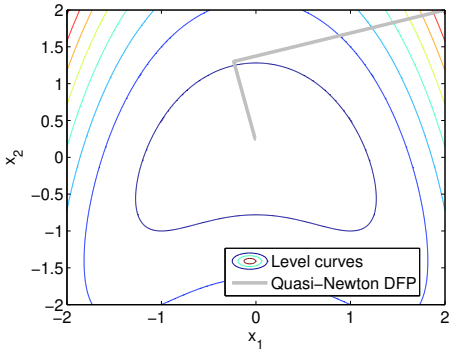


Figure 4.5: Trajectory obtained by a quasi-Newton method.

13. $f(x_1, x_2) = x_1^3 + 6x_1^4 - 3x_2^2 + 2x_2^6$
14. $f(x_1, x_2) = (x_1 + 43x_2)^3 + 3x_1^2 - 5$
15. $f(x_1, x_2) = (x_1 + x_2)^4 - (x_1 + x_2)^2$
16. $f(x_1, x_2) = 33x_1^2 + (x_2 - x_1)^2 + x_1x_2$
17. $f(x_1, x_2) = x_1x_2 + 3x_2 + x_1^2 + x_1^2 + x_2^2$
18. $f(x_1, x_2) = x_1x_2 + x_1^2x_2^2 + 5x_1^2 + 1$
19. $f(x_2, x_4) = x_2^2 + x_4^3 + x_2^6x_4^2$
20. $f(x_1, x_2) = x_1^2 + 2x_1x_2^3 + x_2^2$
21. $f(x_1, x_2) = 3x_1^2 + 2x_1^4 - x_2 + x_2^2 + 1$
22. $f(x_1, x_2) = (x_1 - x_2)^3 + (2x_1 + x_2)^2$
23. $f(x_1, x_2) = x_1^4 + 2x_1^2x_2 + x_2^6 + 6x_1x_2^2 + 3$
24. $f(x_1, x_2) = x_1x_2 + x_2^2 + x_2^4 + 3x_1^4 - 1$
25. $f(x_1, x_2) = 2x_1^2x_2 + 31x_1^4 + 18x_2^2 + 3$
26. $f(x_1, x_2) = 11x_1 + 22x_1^2x_2 + x_2^2 + 31x_1^2$
27. $f(x_1, x_2) = 2x_2^2 + 3x_2^4 + 5x_1^2 + 3$
28. $f(x_1, x_2) = -x_1^3 + x_1^4 + x_2 + x_2^2$
29. $f(x_1, x_2) = x_1^2x_2^2 + 113x_1x_2 - 1$

30. $f(x_1, x_2) = x_1^3 + 6x_1^4 + x_2^2 + x_2^4$
31. $f(x_1, x_2) = x_1^4 + x_2^6 + x_1 x_2 - 8$
32. $f(x_1, x_2) = -10x_2 + x_2^2 + x_2^4 + 3x_1^2 - 1$
33. $f(x_1, x_2) = x_1^2 x_2 + 11x_1^4 + 18x_2^2 - 15$
34. $f(x_1, x_2) = x_1 + 2x_1^2 x_2 + 2x_2^2 + 4x_1^2$
35. $f(x_1, x_2) = x_1^2 x_2 + 5x_2^3 + x_2^4 + 3x_1^2$
36. $f(x_1, x_2) = x_1 x_2 + 3x_2^2 + 4x_2^4 + x_1^2$
37. $f(x_1, x_2) = e^{x_1^2} + x_2^2 - 3 + 2x_2$
38. $f(x_1, x_2) = 2x_1^2 + 3x_1 x_2 + x_2^3 + x_2^2 - 1$
39. $f(x_1, x_2) = (x_1^2 - 1)^2 + x_2^2 - 3x_2 + 1$
40. $f(x_1, x_2) = x_1^2 e^{x_1} + 51x_2 + x_2^4 + 3$
41. $f(x_1, x_2) = 6x_1^2 x_2^2 + 3x_1 x_2 - 1$
42. $f(x_1, x_2) = 4x_1^3 + x_1^4 + x_1 x_2^2 + x_2^4$
43. $f(x_1, x_2) = x_1^4 + x_2 + 3x_1 + 6$
44. $f(x_1, x_2) = x_1^4 + 2x_1^2 x_2 + x_2^2 + 3$
45. $f(x_1, x_2) = e^{x_1^2 - 3} + x_2^2 - 3x_2$
46. $f(x_1, x_2) = x_1 x_2^3 + 2x_1^2 + 2x_2^4 - 5$
47. $f(x_1, x_2) = x_1^2 x_2 + 6x_2^2 - 3x_1 x_2 + 4x_1^4 + 3x_1^2$
48. $f(x_1, x_2, x_3) = 2x_1^2 + 3x_2^2 + x_2 x_3 + 6x_3^2$
49. $f(x_1, x_2) = -2x_1^2 + x_1 x_2^3 - x_2^2$
50. $f(x_1, x_2) = 3x_1^2 - x_1^4 + x_2 + x_2^2 + 1$
51. $f(x_1, x_2) = (x_1 - x_2)^2 + (x_1 + x_2)^2$
52. $f(x_1, x_2) = x_1^2 + 2x_1^2 x_2 + x_2^1 6 + 6x_1 x_2^2 + 3$
53. $f(x_1, x_2) = x_1 x_2 + x_2^2 - x_2^4 + 3x_1^4 + 21$
54. $f(x_1, x_2) = 2x_1^2 x_2 + 25x_1^4 + 9x_2^2 + 13$

Chapter 5

Derivative-free methods

5.1 Introduction

The methods presented in Chapter 4 are first and second order methods, i.e., they make use of the gradient and the Hessian of the objective function. In many cases, the objective function may not be differentiable. In these cases, a zero-order or derivative-free method has to be used. Two such methods are the Nelder-Mead and the Rosenbrock methods.

The Nelder-Mead (also called simplex or amoeba) method, developed by Nelder and Mead in 1965 (see (Rao, 1978)), is based on the iterative modification of a simplex. A simplex is a geometric figure formed by a set of $n + 1$ points in the n-dimensional space. For instance, in 2 dimensions, a simplex is a triangle, while in the 3-dimensional space, a simplex is a tetrahedron. The method replaces one point of the simplex in each iteration, such that the new simplex is gradually moved towards the optimum point. The movement of the simplex is achieved by using four operations: reflection, expansion, contraction, and shrinking. The algorithm stops when all edges of the simplex become smaller than a predefined tolerance or when the simplex degenerates (e.g., in 2 dimensions, the vertices of the triangle become co-linear).

For the two-dimensional case, the details are given in Algorithm 5.1.

It has been recently proven that for the 2-dimensional case, the Nelder-Mead method converges to a local optimum. For higher-dimensions, no convergence results exist yet.

A derivative-free method whose convergence to a local optimum has been proven, is the Rosenbrock method. The Rosenbrock method (also called the method of rotating coordinates) is based on rotating the coordinates such that the first one is oriented towards a locally estimated minimum while the rest are orthogonal to it and are normalized. To rotate the coordinates, the so-called Gramm-Schmidt orthonormalization procedure is used.

The Rosenbrock method is summarized in Algorithm 5.2. The stopping criteria

Algorithm 5.1 Nelder-Mead method for 2 variables

Input: Objective function $f(x, y)$, tolerance ε, initial vertices $V_1,\ V_2,\ V_3$
  while stop criterion not satisfied **do**
    Compute $f(V_1)$, $f(V_2)$, $f(V_3)$ and set the labels B, G, W
    Compute $M = (B + G)/2$ and $R = 2M - W$
    if $f(R) < f(W)$ **then**
      Compute $E = 2R - M$, $f(E)$
      if $f(E) < f(R)$ **then**
        Replace W with E
      else
        Replace W with R
      end if
    else
      Compute $C_1 = (M + W)/2$
      Compute $C_2 = (M + R)/2$
      Choose $C = \text{argmin}_{C_1,C_2}(f(C_1), f(C_2))$
      if $f(C) < f(W)$ **then**
        Replace W with C
      else
        Compute $S = (B + W)/2$
        Replace W with S
        Replace G with M
      end if
    end if
    Set $V_1 = B$
    Set $V_2 = G$
    Set $V_3 = W$
  end while
Output: Minimum with tolerance ε

are those specified in Chapter 4, except for the norm of the gradient.

It should be noted that the Gramm-Schmidt ortho-normalization procedure is numerically unstable and if the new coordinates are not orthogonal, the errors accumulate and the method may diverge.

5.2 Example

Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$, $f(\boldsymbol{x}) = x_1^4 + 2x_1^2x_2 + 2x_2^2 - x_2 + 3$, represented in Figure 5.1. This function has a local minimum in $\begin{pmatrix} 0 & \frac{1}{4} \end{pmatrix}^T$, verifiable using analytical methods. Here, we attempt to find this minimum using the Nelder-Mead and Rosenbrock methods described above.

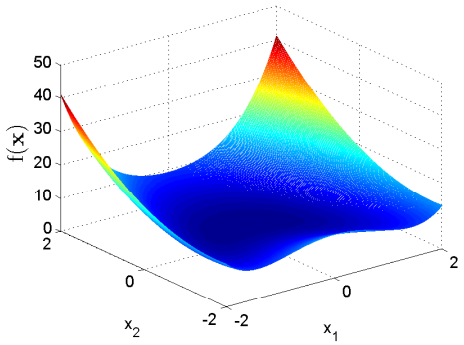


Figure 5.1: Graphical representation of the function $f(\boldsymbol{x}) = x_1^4 + 2x_1^2x_2 + 2x_2^2 - x_2 + 3$.

Let us use the Nelder-Mead method first. The initial points are taken as $[0,\ 0]^T$, $[0,\ 1]^T$, $[1,\ 0]^T$ (represented in Figure 5.2), and the tolerance is chosen as $\varepsilon = 0.0001$.

After the first step, this simplex is contracted, see Figure 5.3, the vertices becoming $[0,\ 0]^T$, $[0,\ 1]^T$, $[0.5,\ 0.25]^T$.

The simplex obtained after 10 steps in presented in Figure 5.4, the vertices becoming $[0.002,\ 0.194]^T$, $[0.037,\ 0.307]^T$, $[-0.057,\ 0.233]^T$.

After 37 steps, the lengths of the edges of the simplex become less than the chosen tolerance ε, the vertices being $[0.12 \cdot 10^{-4},\ 0.25]^T$, $[-0.65 \cdot 10^{-4},\ 0.25]^T$, $[0.3 \cdot 10^{-4},\ 0.25]^T$, which can be well approximated by $[0, 0.25]$, see Figure 5.5.

Let us now see the Rosenbrock method. To compare it with the Nelder-Mead method, we choose the initial point $[0,\ 0]^T$ (one of the vertices used by the Nelder-Mead method) and the same tolerance $\varepsilon = 0.0001$ on the distance between two consecutive points.The initial stepsize is $[0.5,\ 0.5]^T$, $\alpha = 3$, and $\beta = -0.8$. The resulting trajectory is shown in figure 5.6, and the final value is $[-0.00,\ 0.25]^T$. The

Algorithm 5.2 Rosenbrock method

Input: Objective function $f(\boldsymbol{x})$, initial point $\boldsymbol{x}_0$, n orthogonal directions $\boldsymbol{d}_{10}$, $\boldsymbol{d}_{20}$, ..., $\boldsymbol{d}_{n0}$

Input: Initial step lengths $\boldsymbol{s} = [s_{10}\ s_{20}\ \dots s_{n0}]^T$, $\alpha > 1$ and $-1 < \beta < 0$, tolerance ε

Set $k = 1$

while stop criterion not satisfied **do**

Initialize the vector of successful steps for all directions $c = [0\ 0\ \dots 0]$

Initialize flag for oscillation: $oscillation = false$

Initialize a vector to store successes on each direction: $success = [0\ 0\ \dots 0]$

Initialize a vector to store failures on each direction: $fail = [0\ 0\ \dots 0]$

while $oscillation = false$ **do**

for all directions $i = 1, 2, \dots, n$ **do**

if $f(\boldsymbol{x}_k + s_i \boldsymbol{d}_i) \leq f(\boldsymbol{x}_k)$ **then**

Compute $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + s_i \boldsymbol{d}_i$

Set $k \leftarrow k + 1$

Mark a success on direction $\boldsymbol{d}_i$. Set $success(i) = 1$

Add the step length to c_i. Set $c_i \leftarrow c_i + s_i$

Increase the step length. Set $s_i \leftarrow s_i \cdot \alpha$

else

Mark a failure on direction $\boldsymbol{d}_i$. Set $fail(i) = 1$

Decrease the step length. Set $s_i \leftarrow s_i \cdot \beta$

end if

end for

if all $sucess(i) = 1$ and all $fail(i) = 1$ **then**

Set $oscillation = true$

end if

end while

$a_1 = c_1 \boldsymbol{d}_1 + c_2 \boldsymbol{d}_2 + \dots + c_n \boldsymbol{d}_n$

$a_2 = c_2 \boldsymbol{d}_2 + \dots + c_n \boldsymbol{d}_n$

$\dots$

$a_n = c_n \boldsymbol{d}_n$

Compute the new directions using the Gramm-Schmidt procedure from a

$b_1 = a_1$, $\boldsymbol{d}_1 = \frac{b_1}{\|b_1\|}$

$b_2 = a_2 - \frac{a_2^T b_1}{\|b_1\|^2} b_1$, $\boldsymbol{d}_2 = \frac{b_2}{\|b_2\|}$

$b_3 = a_3 - \frac{a_3^T b_1}{\|b_1\|^2} b_1 - \frac{a_3^T b_2}{\|b_2\|^2} b_2$, $\boldsymbol{d}_2 = \frac{b_3}{\|b_3\|}$

$\vdots$

$b_n = a_n - \sum_{i=1}^{n-1} \frac{a_n^T b_i}{\|b_i\|^2} b_i$, $\boldsymbol{d}_n = \frac{b_n}{\|b_n\|}$

end while

Output: Minimum with tolerance ε

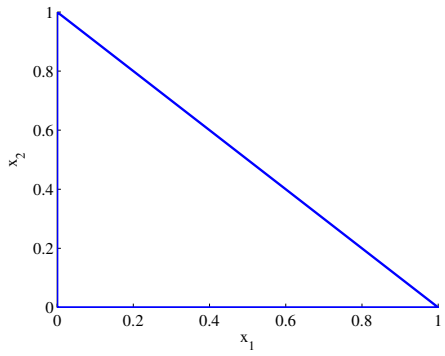


Figure 5.2: Original simplex for the Nelder-Mead method.

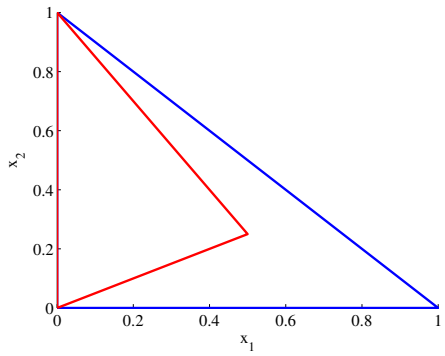


Figure 5.3: Simplex after the first step using the Nelder-Mead method.

Gramm-Schmidt orthonormalization has been performed only once, and 8 successful steps were taken.

For the initial point $[2,\ 2]^T$, tolerance $\varepsilon = 10^{-5}$, initial stepsize $[0.5,\ 0.5]^T$, $\alpha = 3$, $\beta = -0.8$, the same result of $[0.00,\ 0.25]^T$ is obtained, but in this case after 51 successful steps and performing 23 times the Gramm-Schmidt orthonormalization. The trajectory is presented in Figure 5.7.

It has to be noted that although for this particular example both methods found the local minimum, for objective functions with several minima, they may converge to different points.

5.3 Exercises

Consider the following functions $f : \mathbb{R}^n \to \mathbb{R}$. Implement the Nelder-Mead and Rosenbrock methods and find the minima of the functions.

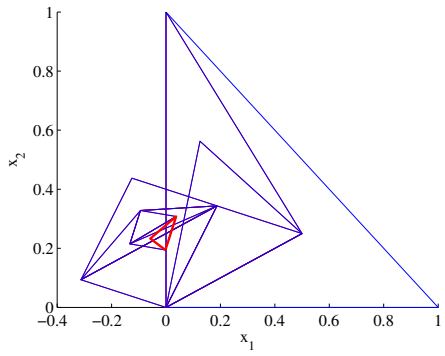


Figure 5.4: Simplex after 10 steps using the Nelder-Mead method.

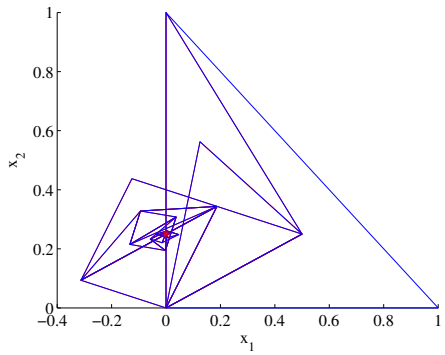


Figure 5.5: Simplex after 37 steps using the Nelder-Mead method.

1. $f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 - 4x_2$
2. $f(x_1, x_2) = 3x_1^2 + 2x_1^4 - x_2 + x_2^2 + 1$
3. $f(x_1, x_2) = e^{x_1^2} + x_2^2 - 3 + 2x_2$
4. $f(x_1, x_2) = (x_1^2 - 1)^2 + x_2^2 - 3x_2 + 1$
5. $f(x_1, x_2) = x_1^2 e^{x_1} + 51x_2 + x_2^4 + 3$
6. $f(x_1, x_2) = x_1^4 + 2x_1^2 x_2 + x_2^2 + 3$
7. $f(x_1, x_2) = e^{x_1^2 - 3} + x_2^2 - 3x_2$
8. $f(x_1, x_2) = x_1 x_2^3 + 2x_1^2 + 2x_2^4 - 5$
9. $f(x_1, x_2) = x_1^2 x_2 + 6x_2^2 - 3x_1 x_2 + 4x_1^4 + 3x_1^2$

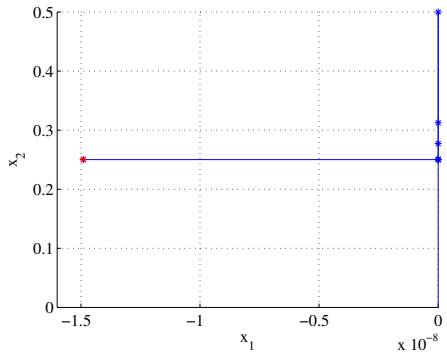


Figure 5.6: Trajectory from $[0,\ 0]^T$ using the Rosenbrock method.

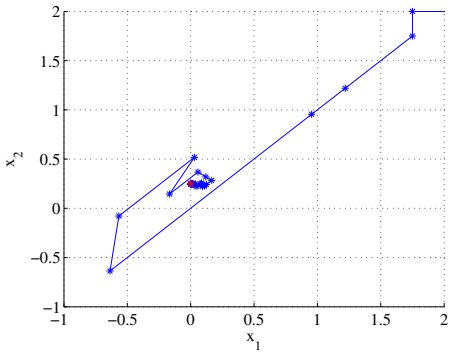


Figure 5.7: Trajectory from $[2,\ 2]^T$ using the Rosenbrock method.

10. $f(x_1, x_2) = x_1^6 + 3x_1x_2 + x_2^2$
11. $f(x_1, x_2) = 18x_1^2 + 20x_2^4 + x_1x_2 + x_2^2$
12. $f(x_1, x_2) = 5x_1^4 + x_1^2x_2 + x_2^2 - 3x_2 + 1$
13. $f(x_1, x_2) = (\cos(2\pi))^{x_1} + x_1x_2 + x_2^2 + x_1^2$
14. $f(x_1, x_2) = x_1^3 + 3x_1^4 + 31x_1x_2 + x_2^2$
15. $f(x_1, x_2) = 6x_1^2x_2^2 + 3x_1x_2 - 1$
16. $f(x_1, x_2) = 4x_1^3 + 6x_1^4 + 3x_1x_2^2 + x_2^4$
17. $f(x_1, x_2) = x_1^4 + 2x_1^2x_2 + x_2^6 + 6x_1x_2^2 + 3$
18. $f(x_1, x_2) = x_1x_2 + x_2^2 + x_2^4 + 3x_1^4 - 1$

19. $f(x_1, x_2) = 2x_1^2x_2 + 31x_1^4 + 18x_2^2 + 3$
20. $f(x_1, x_2) = x_1x_2 + 3x_2^2 + 4x_2^4 + x_1^2$
21. $f(x_1, x_2) = x_1^4 + x_2^2 + 3x_1 + 6$
22. $f(x_1, x_2) = 10x_1^2 + 6x_2^2 + 8x_1^4x_2^4 + 24$
23. $f(x_1, x_2) = 81x_1^2 + 27x_1x_2 + 18x_2^2 + x_2^4 - 9$
24. $f(x_1, x_2) = 5x_1^2 + 3x_1x_2 - x_2^2 + x_2^4$
25. $f(x_1, x_2) = x_1^3 + 6x_1^4 - 3x_2^2 + 2x_2^6$
26. $f(x_1, x_2) = 33x_1^2 + (x_2 - x_1)^2 + x_1x_2$
27. $f(x_1, x_2) = e^{x_1^2} + 3x_1^2 + 1$
28. $f(x_1, x_2) = x_1^4 + 3x_1^2x_2^2 + 5x_2^4 + 15$
29. $f(x_1, x_2) = x_1^6 + x_1^4 + x_1^2x_2^2 + x_2^4 - 3$
30. $f(x_1, x_2) = x_1^2x_2^2 + 6x_1x_2^2 + 15x_2^2 - 1$
31. $f(x_1, x_2) = x_1x_2 + 18x_1^2 + 20x_2^2 + x_2^4$
32. $f(x_1, x_2) = x_1^2 + 38x_1x_2 + 1024x_2^2 + 45x_2^6$
33. $f(x_1, x_2) = x_1x_2^3 + 3x_1^2 + 64x_2^2 + 128x_2^4$
34. $f(x_1, x_2) = x_1^2x_2^2 + 6x_1x_2^2 - 2x_1^2x_2 - 10x_1x_2 + 9x_2^2 - 12x_2 + x_1^2 + 4x_1 + 4$
35. $f(x_1, x_2) = x_1^2 - 2x_1^2x_2 + 4x_1 + x_1^2x_2^2 - 4x_1x_2 + 4$
36. $f(x_1, x_2) = 1 - 2x_1x_2 + 2x_2 + x_1^2x_2^2 - 2x_1x_2^2 + x_2^2$
37. $f(x_1, x_2) = x_1^2x_2^2 - 2x_1x_2^2 + x_2^2$
38. $f(x_1, x_2) = 9 - 18x_1x_2 + 6x_2 + 9x_1^2x_2^2 - 6x_1x_2^2 + x_2^2$
39. $f(x_1, x_2) = 9 - 18x_2 + 6x_1 + 9x_2^2 - 6x_1x_2 + x_1^2$
40. $f(x_1, x_2) = 25 - 10x_2 + 10x_1^2 + x_2^2 - 2x_2x_1^2 + x_1^4$
41. $f(x_1, x_2) = x_1^4 - x_1^3 + x_2^2 - 3x_1$
42. $f(x_1, x_2) = 5x_1^2 + 2x_2^2 + 8x_1^4x_2^4$
43. $f(x_1, x_2) = 8x_1^2 - 2x_1x_2 + 18x_2^2 + x_2^4 - 9$

44. $f(x_1, x_2) = x_1^2 + 13x_1x_2 - 2x_2^2 + x_2^4$
45. $f(x_1, x_2) = x_1 + x_1^4 - 3x_2^2 + 2x_2^6$
46. $f(x_1, x_2) = 3x_1^2 + (2x_2 - x_1)^2 + x_1x_2$
47. $f(x_1, x_2) = x_1^4 + 3x_1^2 + 1 + x_2^4$
48. $f(x_1, x_2) = x_1^6 + x_1^2x_2^2 + 5x_2^4 + 15$
49. $f(x_1, x_2) = x_1^6 + x_1^4 - x_1^2x_2^2 + x_2^4 - 3$
50. $f(x_1, x_2) = x_1^2x_2^4 + 6x_1x_2^2 + 15x_2^2 - 1 + x_1^2$
51. $f(x_1, x_2) = x_1x_2 + 8x_1^2 + 14x_2^2 + x_2^4$
52. $f(x_1, x_2) = x_1^2 + 2x_1x_2 + 10x_2^2 + 45x_2^6$
53. $f(x_1, x_2, x_3) = 29 - 10x_2 + 10x_1 + x_2^2 - 2x_2x_1 + x_1^2 + x_3^2 - 4_x3$
54. $f(x_1, x_2, x_3) = x_2^2 + 2x_2x_1 + x_1^2 + x_3^2 - 4x_3 + 4$
55. $f(x_1, x_2, x_3) = x_2^2 - 2x_2x_1 + x_1^2 + x_3^2 - 4x_3 + 4$
56. $f(x_1, x_2, x_3) = x_2^2 - 2x_2x_1 - 2x_2 + x_1^2 + 2x_1 + 5 + x_3^2 - 4x_3$
57. $f(x_1, x_2, x_3) = x_2^2 - 2x_2x_1 + 2x_2 + x_1^2 - 2x_1 + 2 + x_3^2 - 2x_3$
58. $f(x_1, x_2, x_3) = x_2^2 - 6x_2x_1 + 2x_2 + 9x_1^2 - 6x_1 + 2 + x_3^2 - 2x_3$
59. $f(x_1, x_2, x_3) = x_2^2 - 6x_2x_1 + 4x_2 - 2x_2x_3 + 9x_1^2 - 12x_1 + 6x_1x_3 + 4 - 4x_3 + x_3^2$
60. $f(x_1, x_2, x_3) = x_2^2 - 2x_2x_1 + 4x_2 - 2x_2x_3 + x_1^2 - 4x_1 + 2x_1x_3 + 4 - 4x_3 + x_3^2$
61. $f(x_1, x_2, x_3) = x_2^2 - 2x_2x_1 + 12x_2 - 2x_2x_3 + x_1^2 - 12x_1 + 2x_1x_3 + 36 - 12x_3 + x_3^2$
62. $f(x_1, x_2, x_3) = x_2^2x_1^2 - 2x_2x_1^2 + 12x_2x_1 - 2x_2x_1x_3 + x_1^2 - 12x_1 + 2x_1x_3 + 36 - 12x_3 + x_3^2$

Chapter 6

Linear programming – the simplex method

6.1 Introduction

A linear programming problem is defined as the optimization of a linear function subject to linear constraints, i.e.,

$$\begin{aligned} &\max f(\boldsymbol{x}) = c^T \boldsymbol{x} \quad \text{subject to} \\ &A\boldsymbol{x} \preccurlyeq b \\ &\boldsymbol{x} \succcurlyeq 0 \end{aligned} \tag{6.1}$$

or

$$\begin{aligned} &\min f(\boldsymbol{y}) = b^T \boldsymbol{y} \quad \text{subject to} \\ &A^T \boldsymbol{y} \succcurlyeq c \\ &\boldsymbol{y} \succcurlyeq 0 \end{aligned} \tag{6.2}$$

where $\succcurlyeq$ ($\preccurlyeq$) means that each element of the vector on the left hand side is larger (smaller) than the corresponding element of the vector on the right-hand side. The form (6.1) is called the standard maximization form, while (6.2) is the standard minimization form (Raica, 2009; Rao, 1978).

The linear programming problem has first been recognized by economists in the 1930s while developing methods for the optimal allocation of resources. A solution to a linear programming problem can be found by the simplex method, developed by G. Dantzig in 1947 (Rao, 1978). The method is being used in a large number of applications, such as petroleum refineries, optimal production planning, food processing, metal working industries, etc.

Once a linear programming problem is in the standard maximization form, the simplex method can be applied as follows (Raica, 2009):

1. Inequality constraints (except for those that state that the variables are positive) are converted into equalities by adding non-negative slack variables.

2. An initial basic feasible solution is chosen. A feasible solution is one that satisfies all the constraints.

3. The initial table is written, as follows:

x_{b1}	x_1	x_2	...	x_{n+m}	
x_{b2}	a_{11}	a_{12}	...	$a_{1,n+m}$	b_1
...	...	...	...	...	...
x_{bm}	a_{m1}	a_{m2}	...	$a_{m,n+m}$	b_m
objective	c_1	c_2	...	c_{n+m}	$-f$

   The entries on the left column indicate the basic variables, $a_{i,j}$, b_i, c_i are elements of the matrix A and the vectors b and c, and f is the value of the function for the chosen basic feasible solution.

4. Select the pivot column by choosing the largest positive number from the objective row, excluding $-f$. This step will identify the non-basic variable to enter the basis.

   If all the numbers (excluding $-f$) in the last row are negative or zero the basic solution is the optimal one and the algorithm will stop here.

5. Select the pivot row. This corresponds to the basic variable to leave the basis. The intersection of the pivot row and the pivot column is the pivot element or simply the *pivot*. It must always be a positive number.

   The pivot element is the one that minimizes the ratio b_k/a_{kj} over those rows for which $a_{ij} > 0$.

   If all elements in the pivot column are negative or zero ($a_{kj} \leq 0,\ k = \overline{1,m}$) then the problem is unbounded above (the maximum of the problem is infinity).

6. Perform the pivot operation, when the pivot element is a_{ij}:

   - Divide the pivot row i by the pivot a_{ij}
   - Add $-a_{kj}/a_{ij} \times row(i)$ to row k for each $k \neq i$ (including the objective row). Each element in the rows (non-pivot) will be added by the element in the same row and pivot column divided by the pivot and multiplied by the element in the same column and pivot row.

7. Repeat the operations above until the basic feasible solution is optimal. The algorithm will stop when all the elements in the last row (objective) are negative or zero. The bottom right entry, which is $-f$ will not be included in this test.

8. The optimal value of the objective function is obtained as minus the bottom-right entry of the table, and is expressed in terms of the basic variables.

6.2 Examples

Consider the following linear programming problem:

$\max f(x_1, x_2) = 3x_1 + 3x_2$, subject to

$$\begin{aligned} &9x_1 + 2x_2 \leq 196 \qquad \text{(C1)} \\ &8x_1 + 10x_2 \leq 105 \qquad \text{(C2)} \\ &x_1 \geq 0 \\ &x_2 \geq 0 \end{aligned} \tag{6.3}$$

The constraints and the feasible set are presented in Figure 6.1.

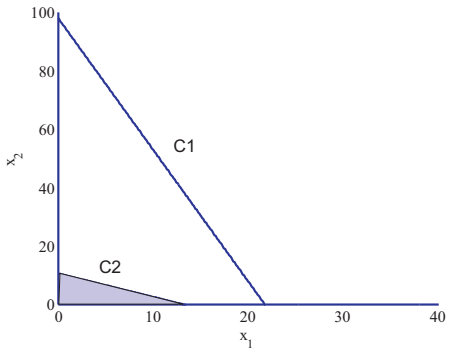


Figure 6.1: Constraints and feasible set for (6.3).

It is quite clear from Figure 6.1 that the constraint (C1) is unnecessary. However, let us proceed with solving the linear programming problem. The problem is in standard maximization form. We will follow the steps described in Section 6.1.

1. The constraints (C1) and (C2) are converted to equalities by introducing the slack variables x_3 and x_4. Consequently, the constraints become

$$\begin{aligned} 9x_1 + 2x_2 + x_3 &= 196 \\ 8x_1 + 10x_2 + x_4 &= 105 \end{aligned}$$

2. We choose the initial basic feasible solution $x_3 = 196$ and $x_4 = 105$.

3. The initial table is:

	x_1	x_2	x_3	x_4	
x_3^*	9	2	1	0	196
x_4^*	8	10	0	1	105
	3	3	0	0	0

4. Select the pivot column: in the last line of the table, both the first and the second columns have the value 3 (maximum of the row), so any of them can be selected. Here, we select the first column. This means that the variable that will enter the basis is x_1.

5. Select the pivot row. For this, we first divide the last column of the table with the pivot column:

	x_1	x_2	x_3	x_4		
x_3^*	9	2	1	0	196	$\frac{196}{9}$
x_4^*	8	10	0	1	105	$\frac{105}{8}$
	3	3	0	0	0	

   Since $\frac{105}{8} < \frac{196}{9}$, the pivot row is the second one, and x_4^* leaves the basis. The pivot element is 8.

6. Pivot operation: the pivot row is divided by the pivot element, and for the other elements on the pivot column zeros are introduced, by adding $-a_{kj}/a_{ij} \times row(i)$ to row k for each $k \neq i$ (including objective row). This means that the pivot row is multiplied by $-\frac{9}{8}$ and added to first row, and is multiplied by $-\frac{3}{8}$ and is added to the last row in the table. The new table is:

	x_1	x_2	x_3	x_4	
x_3^*	0	-9.2500	1	-1.1250	77.8750
x_1^*	1.0000	1.2500	0	0.1250	13.1250
	0	-0.7500	0	-0.3750	-39.3750

7. All the elements in the last row are 0 or negative, thus the algorithm stops. We read the results in terms of the basic variables from the table above:

$$x_3 = 77.8750$$
$$x_1 = 13.125$$
$$\max_{x_1,x_2} f(x_1, x_2) = 39.3750$$

   As can be seen, the maximum of the function is obtained only in terms of x_1 and x_3. From the constraints we obtain: $x_2 = 0$ and $x_4 = 0$.

Consider now the following linear programming problem:

$\min f(y_1, y_2) = 173y_1 + 182y_2$, subject to

$$\begin{aligned} 3y_1 + 2y_2 &\geq 5 \\ 3y_1 + 8y_2 &\geq 2 \\ y_1 &\geq 0 \\ y_2 &\geq 0 \end{aligned}$$

The problem above is in standard minimization form, thus we first rewrite it in standard maximization form:

$\max f(x_1, x_2) = 5x_1 + 2x_2$, subject to

$$\begin{aligned} 3x_1 + 3x_2 &\leq 173 \qquad \text{(C1)} \\ 2x_1 + 8x_2 &\leq 182 \qquad \text{(C2)} \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned} \tag{6.4}$$

The constraints and the feasible set are presented in Figure 6.2.

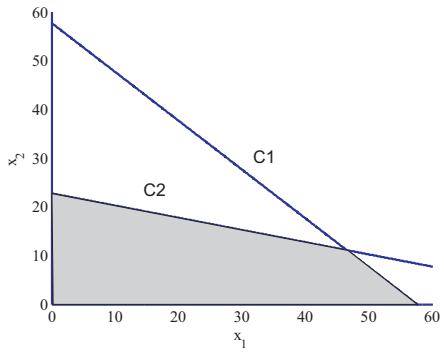


Figure 6.2: Constraints and feasible set for (6.4).

Let us proceed with solving the linear programming problem.

1. The constraints (C1) and (C2) are converted to equalities by introducing the slack variables x_3 and x_4. Consequently, the constraints become

$$\begin{aligned} 3x_1 + 3x_2 + x_3 &= 173 \\ 2x_1 + 8x_2 + x_4 &= 182 \end{aligned}$$

2. We choose the initial basic feasible solution $x_3 = 173$ and $x_4 = 182$.

3. The initial table is:

	x_1	x_2	x_3	x_4	
x_3^*	3	3	1	0	173
x_4^*	2	8	0	1	182
	5	2	0	0	0

4. Select the pivot column: in the last line of the table, the largest value corresponds to the first column. Thus, the first column is the pivot column, and the variable that will enter the basis is x_1.

5. Select the pivot row. For this, we first divide the last column of the table with the pivot column:

	x_1	x_2	x_3	x_4		
x_3^*	3	3	1	0	173	$\frac{173}{3}$
x_4^*	2	8	0	1	182	91
	5	2	0	0	0	

Since $\frac{173}{3} < 91$, the pivot row is the first one, and x_3^* leaves the basis. The pivot element is the 3.

6. Pivot operation. The new table is:

	x_1	x_2	x_3	x_4	
x_1^*	1.0000	1.0000	0.3333	0	57.6667
x_4^*	0	6.0000	-0.6667	1.0000	66.6667
	0	-3.0000	-1.6667	0	-288.3333

7. All the elements in the last row are 0 or negative, thus the algorithm stops. We read the results in terms of the basic variables from the table above:

$$\begin{aligned} x_1 &= 57.6667 \\ x_4 &= 66.6667 \\ \max_{x_1,x_2} f(x_1, x_2) &= 288.33 \end{aligned}$$

As can be seen, the maximum of the function is obtained only in terms of x_1 and x_4. From the constraints we obtain: $x_2 = 0$ and $x_3 = 0$.

6.3 Exercises

Consider the following linear programming problems and solve them using the simplex method.

1. $\max f(x_1, x_2, x_3) = x_1 + x_2 + x_3$, subject to

$$\begin{aligned} x_3 + x_2 &\leq 1 \\ 14x_1 + x_3 &\leq 4 \\ x_1 + x_2 - 2x_3 &= 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned}$$

2. $\max f(x_1, x_2, x_3 = x_1 + x_3$, subject to

$$\begin{aligned} 4x_2 + x_3 &\leq 50 \\ 4x_1 + x_3 &\leq 10 \\ x_1 + x_2 - x_3 &= 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned}$$

3. $\max f(x_1, x_2) = 2x_1 + x_2$, subject to

$$\begin{aligned} x_1 + 4x_2 &\leq 20 \\ 5x_1 + x_2 &\leq 10 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

4. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} x_1 + 4x_2 &\leq 10 \\ 5x_1 + 3x_2 &\leq 10 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

5. $\max f(x_1, x_2) = 6x_1 + 3x_2$, subject to

$$\begin{aligned} 10x_1 + 4x_2 &\leq 10 \\ 5x_1 + 10x_2 &\leq 10 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

6. $\max f(x_1, x_2) = 5x_1 + 3x_2$, subject to

$$\begin{aligned} 6x_1 + 12x_2 &\leq 50 \\ 5x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

7. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} x_1 + 15x_2 &\leq 100 \\ 5x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

8. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} 3x_1 + 15x_2 &\leq 70 \\ 5x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

9. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} 5x_1 + 15x_2 &\leq 100 \\ 5x_1 + 2x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

10. $\max f(x_1, x_2) = 4x_1 + 5x_2$, subject to

$$\begin{aligned} 10x_1 + 15x_2 &\leq 150 \\ 6x_1 + 2x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

11. $\max f(x_1, x_2) = 2x_1 + 3x_2$, subject to

$$\begin{aligned} 10x_1 + 20x_2 &\leq 200 \\ 3x_1 + x_2 &\leq 10 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

12. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} 3x_1 + 20x_2 &\leq 100 \\ 4x_1 + 2x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

13. $\max f(x_1, x_2) = 2x_1 + 3x_2$, subject to

$$\begin{aligned} 33x_1 + 50x_2 &\leq 300 \\ 4x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

14. $\max f(x_1, x_2) = 3x_1 + 3x_2$, subject to

$$\begin{aligned} 18x_1 + 40x_2 &\leq 250 \\ 2x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

15. $\max f(x_1, x_2) = 2x_1 + 2x_2$, subject to

$$\begin{aligned} 20x_1 + 30x_2 &\leq 300 \\ 4x_1 + 3x_2 &\leq 30 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

16. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} 13x_1 + 20x_2 &\leq 150 \\ 40x_1 + 20x_2 &\leq 200 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

17. $\max f(x_1, x_2) = x_1 + x_2$, subject to

$$\begin{aligned} 17x_1 + 31x_2 &\leq 300 \\ 2x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

18. $\max f(x_1, x_2) = 2x_1 + 2x_2$, subject to

$$\begin{aligned} 23x_1 + 31x_2 &\leq 300 \\ 2x_1 + x_2 &\leq 15 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

19. $\max f(x_1, x_2) = 4x_1 + 3x_2$, subject to

$$\begin{aligned} 19x_1 + 27x_2 &\leq 150 \\ 33x_1 + 18x_2 &\leq 200 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

20. $\max f(x_1, x_2) = x_1 + x_2$, subject to

$$\begin{aligned} 42x_1 + 51x_2 &\leq 500 \\ 2x_1 + x_2 &\leq 20 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

21. $\min f(x_1, x_2) = 500x_1 + 20x_2$, subject to

$$\begin{aligned} 42x_1 + 2x_2 &\geq 1 \\ 51x_1 + x_2 &\geq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

22. $\min f(x_1, x_2) = 150x_1 + 200x_2$, subject to

$$\begin{aligned} 19x_1 + 33x_2 &\geq 4 \\ 27x_1 + 18x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

23. $\min f(x_1, x_2) = 300x_1 + 15x_2$, subject to

$$\begin{aligned} 23x_1 + 2x_2 &\geq 2 \\ 31x_1 + x_2 &\geq 2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

24. $\min f(x_1, x_2) = 300x_1 + 20x_2$, subject to

$$\begin{aligned} 17x_1 + 2x_2 &\geq 1 \\ 31x_1 + x_2 &\geq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

25. $\min f(x_1, x_2) = 150x_1 + 200x_2$, subject to

$$\begin{aligned} 13x_1 + 40x_2 &\geq 4 \\ 20x_1 + 20x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

26. $\min f(x_1, x_2) = 300x_1 + 30x_2$, subject to

$$\begin{aligned} 20x_1 + 4x_2 &\geq 2 \\ 30x_1 + 3x_2 &\geq 2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

27. $\min f(x_1, x_2) = 250x_1 + 20x_2$, subject to

$$\begin{aligned} 18x_1 + 2x_2 &\geq 3 \\ 40x_1 + x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

28. $\min f(x_1, x_2) = 300x_1 + 20x_2$, subject to

$$\begin{aligned} 33x_1 + 4x_2 &\geq 2 \\ 50x_1 + x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

29. $\min f(x_1, x_2) = 100x_1 + 20x_2$, subject to

$$\begin{aligned} 3x_1 + 4x_2 &\geq 4 \\ 20x_1 + 2x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

30. $\min f(x_1, x_2) = 200x_1 + 10x_2$, subject to

$$\begin{aligned} 10x_1 + 3x_2 &\geq 2 \\ 20x_1 + x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

31. $\min f(x_1, x_2) = 150x_1 + 20x_2$, subject to

$$\begin{aligned} 10x_1 + 6x_2 &\geq 4 \\ 15x_1 + 2x_2 &\geq 5 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

32. $\min f(x_1, x_2) = 100x_1 + 20x_2$, subject to

$$\begin{aligned} 5x_1 + 5x_2 &\geq 4 \\ 15x_1 + 2x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

33. $\min f(x_1, x_2) = 70x_1 + 20x_2$, subject to

$$\begin{aligned} 3x_1 + 5x_2 &\geq 5 \\ 15x_1 + x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

34. $\min f(x_1, x_2) = 100x_1 + 20x_2$, subject to

$$\begin{aligned} x_1 + 5x_2 &\geq 4 \\ 15x_1 + x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

35. $\min f(x_1, x_2) = 50x_1 + 20x_2$, subject to

$$\begin{aligned} 6x_1 + 5x_2 &\geq 5 \\ 12x_1 + 1x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

36. $\min f(x_1, x_2) = 10x_1 + 10x_2$, subject to

$$\begin{aligned} 10x_1 + 5x_2 &\geq 6 \\ 4x_1 + 10x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

37. $\min f(x_1, x_2) = 10x_1 + 10x_2$, subject to

$$\begin{aligned} x_1 + 5x_2 &\geq 4 \\ 4x_1 + 3x_2 &\geq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

38. $\min f(x_1, x_2) = 20x_1 + 10x_2$, subject to

$$\begin{aligned} x_1 + 5x_2 &\geq 2 \\ 4x_1 + x_2 &\geq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

39. $\min f(x_1, x_2) = 50x_1 + 10x_2$, subject to

$$\begin{aligned} 2x_1 + 5x_2 &\geq 2 \\ 6x_1 + x_2 &\geq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

40. $\min f(x_1, x_2) = x_1 + 2x_2$, subject to

$$\begin{aligned} 2x_1 + 8x_2 &\geq 1 \\ 3x_1 + x_2 &\geq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

41. $\max f(x_1, x_2) = 175x_1 + 78x_2$, subject to

$$\begin{aligned} 202x_1 + 143x_2 &\leq 243 \\ 393x_1 + 137x_2 &\leq 442 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

42. $\max f(x_1, x_2) = 494x_1 + 68x_2$, subject to

$$\begin{aligned} 180x_1 + 7x_2 &\leq 21 \\ 81x_1 + 216x_2 &\leq 176 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

43. $\max f(x_1, x_2) = 116x_1 + 192x_2$, subject to

$$\begin{aligned} 45x_1 + 183x_2 &\leq 146 \\ 100x_1 + 8x_2 &\leq 151 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

44. $\max f(x_1, x_2) = 59x_1 + 30x_2$, subject to

$$\begin{aligned} 76x_1 + 187x_2 &\leq 166 \\ 104x_1 + 27x_2 &\leq 187 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

45. $\max f(x_1, x_2) = 94x_1 + 150x_2$, subject to

$$\begin{aligned} 41x_1 + 145x_2 &\leq 18 \\ 147x_1 + 28x_2 &\leq 4 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

46. $\max f(x_1, x_2) = 85x_1 + 166x_2$, subject to

$$\begin{aligned} 63x_1 + 8x_2 &\leq 11 \\ 20x_1 + 46x_2 &\leq 11 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

47. $\max f(x_1, x_2) = 174x_1 + 73x_2$, subject to

$$\begin{aligned} 116x_1 + 22x_2 &\leq 32 \\ 92x_1 + 186x_2 &\leq 141 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

48. $\max f(x_1, x_2) = 117x_1 + 151x_2$, subject to

$$\begin{aligned} 35x_1 + 129x_2 &\leq 96 \\ 100x_1 + 50x_2 &\leq 196 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

49. $\max f(x_1, x_2) = 148x_1 + 188x_2$, subject to

$$\begin{aligned} 18x_1 + 131x_2 &\leq 87 \\ 102x_1 + 108x_2 &\leq 124 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

50. $\max f(x_1, x_2) = 84x_1 + 121x_2$, subject to

$$\begin{aligned} 143x_1 + 136x_2 &\leq 145 \\ 18x_1 + 82x_2 &\leq 83 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

51. $\max f(x_1, x_2) = 30x_1 + 17x_2$, subject to

$$\begin{aligned} 22x_1 + 149x_2 &\leq 138 \\ 97x_1 + 5x_2 &\leq 43 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

52. $\max f(x_1, x_2) = 99x_1 + 13x_2$, subject to

$$\begin{aligned} 64x_1 + 4x_2 &\leq 158 \\ 41x_1 + 110x_2 &\leq 164 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

53. $\max f(x_1, x_2) = 95x_1 + 87x_2$, subject to

$$\begin{aligned} 46x_1 + 26x_2 &\leq 98 \\ 3x_1 + 52x_2 &\leq 160 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

54. $\max f(x_1, x_2) = 192x_1 + 188x_2$, subject to

$$\begin{aligned} 97x_1 + 186x_2 &\leq 88 \\ 195x_1 + 134x_2 &\leq 81 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Chapter 7

Quadratic programming – the active set method

7.1 Introduction

A quadratic programming problem is an optimization problem where the objective function is quadratic and the constraints are linear:

$$\begin{aligned} &\min f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} + c^T\boldsymbol{x} \\ &A_1\boldsymbol{x} = b_1 \\ &A_2\boldsymbol{x} \preccurlyeq b_2 \end{aligned}$$

with $f : \mathbb{R}^n \to \mathbb{R}$, $Q \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, n being the number of variables.

Here we assume that $Q = Q^T \geq 0$ (positive semi-definite), so that the problem to be solved is a convex quadratic problem. We have m equality constraints corresponding to $A_1\boldsymbol{x} = b_1$ and p inequality constraints, corresponding to $A_2\boldsymbol{x} \preccurlyeq b_2$. To solve the above problem, the method of Lagrange multipliers can be used. Recall that the Lagrangean of the problem can be written as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} + c^T\boldsymbol{x} + \sum_{i=1}^{m} \lambda_i(\boldsymbol{a}_i\boldsymbol{x} - b_i) + \sum_{j=m+1}^{m+p} \lambda_j(\boldsymbol{a}_j\boldsymbol{x} - b_j) \qquad (7.1)$$

where $\boldsymbol{a}_i$ ($\boldsymbol{a}_j$) denotes the ith ($j-m$th) row of the matrix A_1 (A_2) and $\boldsymbol{\lambda}$ is the vector of $m+p$ Lagrange multipliers.

We can write the Karush-Kuhn-Tucker (KKT) conditions for this problem:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{x}} &= 0 && (7.2)\\
\boldsymbol{a}_i^T \boldsymbol{x} &= b_i, \;\; i = 1, \ldots, m && (7.3)\\
\boldsymbol{a}_j^T \boldsymbol{x} &\leq b_j, \;\; j = m+1, \ldots, m+p && (7.4)\\
\lambda_j(\boldsymbol{a}_j^T \boldsymbol{x} - b_j) &= 0, \;\; j = m+1, \ldots, m+p && (7.5)\\
\lambda_j &\geq 0, \;\; j = m+1, \ldots, m+p && (7.6)\\
\lambda_i &\quad , \;\; i = 1, \ldots, m \text{ unrestricted in sign} && (7.7)
\end{aligned}$$

Since the objective function is convex and the constraints are linear, the KKT conditions are both necessary and sufficient for a point x^* to be an optimum.

One method to solve a convex quadratic problem is the active set method. An active set is the set of constraints that are active (i.e., they are satisfied as equalities) at a certain point. Let us assume now that there are no equality constraints, i.e., the constraints are $\bar{A}\boldsymbol{x} \preccurlyeq \bar{b}$. The algorithm for the active set method is described in Algorithm 7.1. In the method below $\boldsymbol{a}_i$ denotes the ith row of the constraint matrix $\bar{A}$ and λ_i are the Lagrange multipliers corresponding to the active constraints. The number of active constraint may vary throughout the steps from 0 to m, m being the number of constraints.

7.2 Example

Consider the following convex quadratic programming problem:

$$\min f(x_1, x_2) = \frac{1}{2}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} 8 & 5.5 \\ 5.5 & 13 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} -7 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

subject to

$$\begin{aligned}
x_1 + 10x_2 &\leq 25 && \text{C1}\\
x_1 - x_2 &\leq 3 && \text{C2}\\
-x_1 + 3x_2 &\leq 4 && \text{C3}\\
-x_1 - 3x_2 &\leq -6 && \text{C4}
\end{aligned} \qquad (7.13)$$

i.e., $Q = \begin{pmatrix} 8 & 5.5 \\ 5.5 & 13 \end{pmatrix}$, $c = \begin{pmatrix} -7 \\ 2 \end{pmatrix}$, $\bar{A} = \begin{pmatrix} 1 & 10 \\ 1 & -1 \\ -1 & 3 \\ -1 & -3 \end{pmatrix}$, $b = \begin{pmatrix} 25 \\ 3 \\ 4 \\ -6 \end{pmatrix}$.

The constraints (7.13) and the resulting feasible set, together with the contour plot of the objective function are illustrated in Figure 7.1.

Algorithm 7.1 Active set method

Input: Parameters of the quadratic objective function: Q and c

Input: Constraints: $\bar{A}$ and $\bar{b}$, initial feasible point $\boldsymbol{x}_0$, initial working set W_0

Compute the gradient at the current point: $\boldsymbol{g}_0 = Q\boldsymbol{x}_0 + c$

Compute the matrix A having the rows $\boldsymbol{a}_i$, $i \in W_k$

Solve the linear system

$$\begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_0 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g}_0 \\ 0 \end{pmatrix} \tag{7.8}$$

Set $k = 0$

while (not all $\lambda_i \geq 0$) or ($\boldsymbol{d}_k \neq 0$) **do**

if $\boldsymbol{d}_k = 0$ **then**

Check optimality:

if all $\lambda_i \geq 0$ **then**

Stop and return the current point $\boldsymbol{x}_k$

else

Find $j = \text{argmin}_{j \in W_k}(\lambda_j)$

Remove constraint j from the working set W_k

Keep the same point for the next step: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$

end if

else

Compute the step length α_k from:

$$\alpha_k = \min_{i \notin W_k,\, \boldsymbol{a}_i \boldsymbol{d}_k > 0} \left(1, \frac{b_i - \boldsymbol{a}_i \boldsymbol{x}_k}{\boldsymbol{a}_i \boldsymbol{d}_k} \right) \tag{7.9}$$

Compute the new point: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$

if $\alpha_k < 1$ **then**

Find the blocking constraint with index

$$i_b = \text{argmin}_{i \notin W_k,\, \boldsymbol{a}_i \boldsymbol{d}_k > 0} \left(1, \frac{b_i - \boldsymbol{a}_i \boldsymbol{x}_k}{\boldsymbol{a}_i \boldsymbol{d}_k} \right) \tag{7.10}$$

Add the constraint i_b to the working set W_k

end if

end if

Set $k \leftarrow k + 1$

Compute the gradient of the objective function at the current point:

$$\boldsymbol{g}_k = Q\boldsymbol{x}_k + c \tag{7.11}$$

Compute the matrix A having the rows $\boldsymbol{a}_i$, $i \in W_k$

Solve the linear system

$$\begin{pmatrix} Q & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_k \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g}_k \\ 0 \end{pmatrix} \tag{7.12}$$

end while

Output: Minimum point

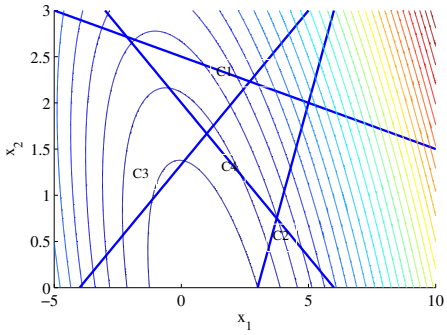


Figure 7.1: The constraints (7.13) and the feasible set.

As can be seen, the unconstrained minimum of the objective function is outside the feasible set. Let us proceed with the active set method, as described by Algorithm 7.1.

A feasible initial point can be chosen e.g., as the intersection of C1 and C3, $x_{10} = \frac{35}{13}$, $x_{20} = \frac{29}{13}$. The corresponding working set is $W_0 = \{1, 3\}$, as C1 and C3 are active at this point. The gradient at this point is[1] $\begin{pmatrix} 26.8 \\ 45.8 \end{pmatrix}$, and the linear matrix equation

$$\begin{pmatrix} 8 & 5.5 & 1 & -1 \\ 5.5 & 13 & 10 & 3 \\ 1 & 10 & 0 & 0 \\ -1 & 3 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -26.80 \\ -45.80 \\ 0 \\ 0 \end{pmatrix}$$

has to be solved. Since there are 2 active constraints, 2 Lagrange multipliers have to be computed. The solution is

$$\boldsymbol{d}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{\lambda} = \begin{pmatrix} -9.71 \\ 17.10 \end{pmatrix}$$

As can be seen, the new directions are 0, but one of the Lagrange multipliers is negative. Thus, the corresponding constraint (C1) is removed from the working set, which becomes $W(1) = \{3\}$. The new point is the same as the previous.

The new linear matrix equation to be solved is:

$$\begin{pmatrix} 8 & 5.5 & 1 \\ 5.5 & 13 & 3 \\ -1 & 3 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_2 \\ \lambda \end{pmatrix} = \begin{pmatrix} -26.80 \\ -45.80 \\ 0 \end{pmatrix}$$

[1] Values are truncated to two decimal places.

with only one active constraint. The solution is

$$\boldsymbol{d}_2 = \begin{pmatrix} -3.20 \\ -1.07 \end{pmatrix}$$
$$\lambda = \begin{pmatrix} -4.75 \end{pmatrix}$$

The direction is no longer 0, therefore we move along the new direction. We compute the stepsize α_k as

$$\alpha_k = \min_{i \notin W_k,\, \boldsymbol{a}_i^T \boldsymbol{d}_k > 0} \left(1, \frac{b_i - \boldsymbol{a}_i^T \boldsymbol{x}_k}{\boldsymbol{a}_i^T \boldsymbol{d}_k} \right)$$

The last constraint (C4) blocks the direction, with a $\alpha = 0.52$. Therefore, the new point is $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha \boldsymbol{d}_2 = \begin{pmatrix} 1 \\ 1.66 \end{pmatrix}$ and the new working set becomes $W = \{3, 4\}$. The new matrix equation is

$$\begin{pmatrix} 8 & 5.5 & -1 & -1 \\ 5.5 & 13 & 3 & -3 \\ -1 & 3 & 0 & 0 \\ -1 & -3 & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_3 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -10.16 \\ -29.16 \\ 0 \\ 0 \end{pmatrix}$$

where $\boldsymbol{g}_1 = \begin{pmatrix} -10.16 \\ -29.16 \end{pmatrix}$ is the gradient of the function in the new point. The solution of the matrix equation above is

$$\boldsymbol{d}_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$
$$\boldsymbol{\lambda} = \begin{pmatrix} 0.22 \\ 9.94 \end{pmatrix}$$

The directions are 0 and both λs are positive, thus the algorithm stops. The minimum point is $\boldsymbol{x}^* = \begin{pmatrix} 1 \\ 1.66 \end{pmatrix}$. The function's value in this point is $f(\boldsymbol{x}) = 27.55$.

7.3 Exercises

Solve the following quadratic programming problems:

$\min f(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T Q \boldsymbol{x} + c^T \boldsymbol{x}$ *subject to*
$\bar{A} \boldsymbol{x} \preceq \bar{b}$

with Q, c, $\bar{A}$, and $\bar{b}$ given as

1.

$$Q = \begin{pmatrix} 1 & 0 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 2 \\ -4 & -1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

2.

$$Q = \begin{pmatrix} 1 & 4 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 2 \\ -3 & -0.5 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

3.

$$Q = \begin{pmatrix} 1 & 2 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 2 \\ -2 & -0.5 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

4.

$$Q = \begin{pmatrix} 1 & 2 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -2 \\ -2 & .5 \\ 1 & 2 \\ 2 & 1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

5.

$$Q = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -2 \\ -4 & .5 \\ 1 & 3 \\ 2 & 1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

6.

$$Q = \begin{pmatrix} 1 & 3 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ -4 & 5 \\ 1 & 3 \\ -2 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \end{pmatrix}$$

7.

$$Q = \begin{pmatrix} 1 & 3 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} 4 \\ -4 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ -4 & 5 \\ 1 & 3 \\ -2 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \end{pmatrix}$$

8.

$$Q = \begin{pmatrix} 1 & 3 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -4 \\ -6 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ -4 & 5 \\ 1 & 3 \\ -2 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \end{pmatrix}$$

9.

$$Q = \begin{pmatrix} 1 & 0 \\ 1 & 6 \end{pmatrix} \qquad c = \begin{pmatrix} -4 \\ -2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ -2 & 3 \\ 1 & 3 \\ -2 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \end{pmatrix}$$

10.

$$Q = \begin{pmatrix} 1 & 0 \\ 1 & 4 \end{pmatrix} \qquad c = \begin{pmatrix} -6 \\ -6 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -5 \\ -2 & 3 \\ 1 & 3 \\ -2 & 1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

11.

$$Q = \begin{pmatrix} 2 & 0 \\ 3 & 10 \end{pmatrix} \qquad c = \begin{pmatrix} -6 \\ -6 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -5 \\ -2 & 3 \\ 1 & 3 \\ -2 & 1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

12.

$$Q = \begin{pmatrix} 1 & 2 \\ 3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.66 \\ 1 & 2 \\ -1 & 0.66 \\ -1 & -2 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.33 \\ 4 \\ 1.33 \\ 4 \end{pmatrix}$$

13.

$$Q = \begin{pmatrix} 1 & 2 \\ 3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ 1 & 2 \\ -1 & 0.5 \\ -1 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1 \\ 4 \\ 1 \\ 4 \end{pmatrix}$$

14.

$$Q = \begin{pmatrix} 1 & 2 \\ 3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.41 \\ 1 & 2 \\ -1 & 0.5 \\ -1 & -1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.58 \\ 4 \\ 1 \\ 4 \end{pmatrix}$$

15.

$$Q = \begin{pmatrix} 1 & 2 \\ 3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.62 \\ 1 & 0.16 \\ -1 & 0.28 \\ -1 & -0.16 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.37 \\ 2.16 \\ 1 \\ 1 \end{pmatrix}$$

16.

$$Q = \begin{pmatrix} 1 & 0 \\ 4 & 8 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.35 \\ 1 & 0.16 \\ -1 & 0.33 \\ -1 & -0.07 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 11.64 \\ 2.16 \\ 1.33 \\ 0.92 \end{pmatrix}$$

17.

$$Q = \begin{pmatrix} 1 & 3 \\ 0 & 8 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.35 \\ 1 & 0.16 \\ -1 & 0.33 \\ -1 & -0.07 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 11.64 \\ 2.16 \\ 1.33 \\ 0.92 \end{pmatrix}$$

18.

$$Q = \begin{pmatrix} 1 & 3 \\ 0 & 8 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ 3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.5 \\ 1 & 0.33 \\ -1 & 2 \\ -1 & -0.07 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2.5 \\ 1.66 \\ 3 \\ 0.92 \end{pmatrix}$$

19.

$$Q = \begin{pmatrix} 2 & 3 \\ 0 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} 5 \\ 3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1.25 \\ 1 & 0.5 \\ -1 & 2 \\ -1 & -0.16 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 3 \\ 0.83 \end{pmatrix}$$

20.

$$Q = \begin{pmatrix} 2 & 3 \\ 0 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} 5 \\ 3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ 1 & 0.12 \\ -1 & 0.6 \\ -1 & -0.4 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 2 \\ 3.8 \\ 0.8 \end{pmatrix}$$

21.

$$Q = \begin{pmatrix} 2 & 1 \\ 0 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.16 \\ 1 & 0.16 \\ -1 & 0.42 \\ -1 & -0.18 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.66 \\ 2.33 \\ 2.42 \\ 1.81 \end{pmatrix}$$

22.

$$Q = \begin{pmatrix} 2 & 0 \\ 3 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ 3 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.22 \\ 1 & 0.14 \\ -1 & 0.42 \\ -1 & -0.22 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.77 \\ 2.14 \\ 2.42 \\ 1.77 \end{pmatrix}$$

23.

$$Q = \begin{pmatrix} 1 & -1 \\ -3 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.2 \\ 1 & 0.16 \\ -1 & 0.42 \\ -1 & -0.22 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.6 \\ 2.33 \\ 2.42 \\ 1.77 \end{pmatrix}$$

24.

$$Q = \begin{pmatrix} 1 & -2 \\ -3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.18 \\ 1 & 1 \\ -1 & 0.42 \\ -1 & -0.4 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 0.72 \\ 9 \\ 2.42 \\ 1.6 \end{pmatrix}$$

25.

$$Q = \begin{pmatrix} 1 & -2 \\ -3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.28 \\ 1 & 0.33 \\ -1 & 0.6 \\ -1 & -0.4 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 0.57 \\ 3.66 \\ 3.8 \\ 0.8 \end{pmatrix}$$

26.

$$Q = \begin{pmatrix} 1 & 2 \\ -3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.13 \\ 1 & 0.2 \\ -1 & 0.6 \\ -1 & -0.13 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.6 \\ 2.6 \\ 3.8 \\ 1.6 \end{pmatrix}$$

27.

$$Q = \begin{pmatrix} 1 & 2 \\ -3 & 7 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.13 \\ 1 & 0.2 \\ -1 & 0.6 \\ -1 & -0.13 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.6 \\ 2.6 \\ 3.8 \\ 1.6 \end{pmatrix}$$

28.

$$Q = \begin{pmatrix} 1 & 1 \\ -3 & 5 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.13 \\ 1 & 0.2 \\ -1 & 0.6 \\ -1 & -0.13 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.6 \\ 2.6 \\ 3.8 \\ 1.6 \end{pmatrix}$$

29.

$$Q = \begin{pmatrix} 1 & 4 \\ 3 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.13 \\ 1 & 0.2 \\ -1 & 0.6 \\ -1 & -0.13 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.6 \\ 2.6 \\ 3.8 \\ 1.6 \end{pmatrix}$$

30.

$$Q = \begin{pmatrix} 1 & 4 \\ 3 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.26 \\ 1 & 0.2 \\ -1 & 0.8 \\ -1 & -0.06 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2.2 \\ 3.6 \\ 4.4 \\ 1.8 \end{pmatrix}$$

31.

$$Q = \begin{pmatrix} 1 & 4 \\ 3 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.5 \\ 1 & 0.2 \\ -1 & 0.8 \\ -1 & -0.75 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.5 \\ 3.6 \\ 4.4 \\ -0.25 \end{pmatrix}$$

32.

$$Q = \begin{pmatrix} 1 & 4 \\ 3 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -1 \\ 1 & 0.6 \\ -1 & 1 \\ -1 & -0.6 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 2 \\ 6.8 \\ 6 \\ -0.4 \end{pmatrix}$$

33.

$$Q = \begin{pmatrix} 1 & 4 \\ 1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.4 \\ 1 & 0.25 \\ -1 & 1 \\ -1 & -0.6 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.4 \\ 4 \\ 6 \\ -0.4 \end{pmatrix}$$

34.

$$Q = \begin{pmatrix} 1 & 4 \\ 1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} 5 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.28 \\ 1 & 0.25 \\ -1 & 1 \\ -1 & -0.42 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.85 \\ 4 \\ 6 \\ 0.28 \end{pmatrix}$$

35.

$$Q = \begin{pmatrix} 1 & 4 \\ 1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} 25 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.28 \\ 1 & 0.25 \\ -1 & 1 \\ -1 & -0.42 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.85 \\ 4 \\ 6 \\ 0.28 \end{pmatrix}$$

36.

$$Q = \begin{pmatrix} 1 & -4 \\ -1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} 3 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.25 \\ 1 & 0.66 \\ -1 & 0.75 \\ -1 & -0.42 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.75 \\ 6.33 \\ 5 \\ 0.28 \end{pmatrix}$$

37.

$$Q = \begin{pmatrix} 1 & -4 \\ -1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -3 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.4 \\ 1 & 0.66 \\ -1 & 0.75 \\ -1 & -0.11 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1 \\ 6.33 \\ 5 \\ 1.55 \end{pmatrix}$$

38.

$$Q = \begin{pmatrix} 1 & -4 \\ -1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -15 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.3636 \\ 1 & 1 \\ -1 & 1 \\ -1 & -0.1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 0.81 \\ 9 \\ 7 \\ 1.5 \end{pmatrix}$$

39.

$$Q = \begin{pmatrix} 1 & -4 \\ -1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -15 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.28 \\ 1 & 1 \\ -1 & 4 \\ -1 & -0.5 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.28 \\ 9 \\ 31 \\ -0.5 \end{pmatrix}$$

40.

$$Q = \begin{pmatrix} 1 & -4 \\ -1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} -10 \\ -1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -0.25 \\ 1 & 3 \\ -1 & 3 \\ -1 & -0.5 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.25 \\ 24 \\ 24 \\ -0.5 \end{pmatrix}$$

41.

$$Q = \begin{pmatrix} 8 & 5 \\ 6 & 13 \end{pmatrix} \qquad c = \begin{pmatrix} -1 \\ -8 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 10 \\ -1 & 1.33 \\ 1 & -3 \\ -1 & -3 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 82 \\ -2.66 \\ -4 \\ -26 \end{pmatrix}$$

42.

$$Q = \begin{pmatrix} 6 & 10 \\ 9 & 19 \end{pmatrix} \qquad c = \begin{pmatrix} -5 \\ -17 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -12 \\ 1 & 0.45 \\ -1 & 0 \\ 1 & -1.4 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} -23 \\ 14.36 \\ -8 \\ -1.8 \end{pmatrix}$$

43.

$$Q = \begin{pmatrix} 10 & 1 \\ 7 & 18 \end{pmatrix} \qquad c = \begin{pmatrix} 0 \\ -8 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} -1 & 2 \\ 1 & -0.57 \\ -1 & -5 \\ 1 & 2.25 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 20 \\ 1.42 \\ -46 \\ 26.75 \end{pmatrix}$$

44.

$$Q = \begin{pmatrix} 14 & 6 \\ 0 & 12 \end{pmatrix} \qquad c = \begin{pmatrix} -3 \\ 1 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 0.36 \\ 1 & 3.66 \\ -1 & 1.3 \\ -1 & -2 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 5 \\ 41.33 \\ -1.75 \\ -5 \end{pmatrix}$$

45.

$$Q = \begin{pmatrix} 8 & 5 \\ 4 & 17 \end{pmatrix} \qquad c = \begin{pmatrix} 12 \\ 8 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 2 \\ -1 & 0.66 \\ -1 & 0.09 \\ 1 & 0.84 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 17 \\ -1 \\ -2.72 \\ 15.85 \end{pmatrix}$$

46.

$$Q = \begin{pmatrix} 6 & 0 \\ 5 & 20 \end{pmatrix} \qquad c = \begin{pmatrix} -2 \\ -6 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 2 \\ 1 & 0.33 \\ 1 & -0.3750 \\ 1 & 0 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 26 \\ 9.33 \\ 8.6250 \\ 12 \end{pmatrix}$$

47.

$$Q = \begin{pmatrix} 13 & 2 \\ 3 & 20 \end{pmatrix} \qquad c = \begin{pmatrix} 10 \\ 0 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & -3 \\ 1 & 0.57 \\ -1 & -2 \\ -1 & 1 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} -39 \\ 11 \\ -21 \\ 20 \end{pmatrix}$$

48.

$$Q = \begin{pmatrix} 14 & 5 \\ 1 & 15 \end{pmatrix} \qquad c = \begin{pmatrix} 3 \\ -8 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} -1 & 0.75 \\ 1 & 1.33 \\ -1 & -0.2 \\ -1 & 4 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 1.25 \\ 21.66 \\ -5.8 \\ 11 \end{pmatrix}$$

49.

$$Q = \begin{pmatrix} 14 & 7 \\ 6 & 17 \end{pmatrix} \qquad c = \begin{pmatrix} -7 \\ 2 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} -1 & 0.25 \\ -1 & -1.33 \\ 1 & 1.5 \\ -1 & 0 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} -8.75 \\ 1 \\ 33 \\ -12 \end{pmatrix}$$

50.

$$Q = \begin{pmatrix} 12 & 6 \\ 7 & 13 \end{pmatrix} \qquad c = \begin{pmatrix} -17 \\ 8 \end{pmatrix} \qquad \bar{A} = \begin{pmatrix} 1 & 7 \\ -1 & -1.1 \\ 1 & 0.2 \\ -1 & 2 \end{pmatrix} \qquad \bar{b} = \begin{pmatrix} 86 \\ -15.2 \\ 13.4 \\ 13 \end{pmatrix}$$

Appendix A

Introduction to MATLAB

A.1 Introduction

MatLab (The Mathworks Inc.) is a commercial "**Mat**rix **Lab**oratory" package which operates as an interactive programming environment for scientific and engineering calculations.

Matlab is a command-driven, interactive language, aimed at solving mathematical problems involving vectors and matrices. The only data structure which Matlab uses is a non-dimensional matrix (or array), the dimensions being adjusted automatically by Matlab as required.

A.2 Statements and variables

Statements have the form:

```
>> variable = expression
```

The command prompt is represented by two right arrows "≫". Equality "=" implies the assignment of the expression to the variable.

The assignment of value 1 to the variable a is executed after the enter key is pressed.

```
>> a = 1
a =
     1
```

The value of the variable is automatically displayed after the statement is executed. If the statement is followed by a semicolon ";" the output is suppressed.

```
>> a = 1;
```

The usual mathematical operators can be used in expressions. The common operators are "+"(addition), "-"(subtraction), "\" (division), "$\star$" (multiplication), "^" (power). The order of arithmetic operations can be altered by using parentheses.

Matlab can be used in "calculator mode". When the variable name and "=" are omitted from an expression, the result is assigned to the generic variable *ans*.

```
>> 3.7*3
ans =
   11.1000
```

A.3 Entering vectors and matrices

A row (line) vector can be created by entering each element (separated by space or comma) between brackets.

```
>> a = [1 2 3 4 5 6 9 8 7]
```

Matlab returns:

```
   a =
       1   2   3   4   5   6   9   8   7
```

A vector with elements evenly spaced between 0 and 20 in increments of 2 (this method is frequently used to create a time or index vector) can be created as follows:

```
>> t = 0:2:20
   t =
           0   2   4   6   8   10   12   14   16   18   20
```

Individual items within the vector can be referenced. To change the fifth element in the t vector:

```
>> t(5) = 23
   t =
           0   2   4   6   23   10   12   14   16   18   20
```

Suppose that we want to add 2 to each of the elements in vector a:

```
>> b = a + 2
     b =
           3   4   5   6   7   8   11   10   9
```

Adding two vectors of the same length:

```
>> c = a + b
     c =
          4   6   8   10   12   14   20   18   16
```

Subtraction of vectors of the same length works exactly the same way.

Entering matrices into Matlab is done by entering each row separated by a semicolon " ;" or a return:

```
>> B = [1 2 3 4;5 6 7 8;9 10 11 12]
   B =
          1    2    3     4
          5    6    7     8
          9   10   11    12

>> B = [ 1  2  3  4
       5  6  7  8
       9 10 11 12]

     B =
          1    2    3     4
          5    6    7     8
          9   10   11    12
```

Matrices in Matlab can be manipulated in many ways. The transpose of a matrix is obtained by using the " ′ " key:

```
>> C = B'
   C =
          1   5    9
          2   6   10
          3   7   11
          4   8   12
```

It should be noted that if C is complex, the apostrophe results in the complex conjugate transpose. To obtain the transpose, use ".′" (the two commands are the same if the matrix is not complex). Now we might multiply the two matrices B and C. Remember that order matters when multiplying matrices.

```
>> D = B * C
   D =
         30    70   110
         70   174   278
```

```
        110   278   446
>> D  = C * B
   D =
        107   122   137   152
        122   140   158   176
        137   158   179   200
        152   176   200   224
```

Element-wise multiplication is obtained using the "$.*$" operator (for matrices of the same sizes).

```
>> E = [1 2;3 4], F = [2 3;4 5], G = E .* F
     E =
          1   2
          3   4
     F =
          2   3
          4   5
     G =
          2   6
         12  20
```

Square matrices can also be raised to a given integer power.

```
>> E^3
     ans =
         37    54
         81   118
```

Element-wise power is obtained by using "`.^`"

```
>> E.^3
     ans =
          1     8
         27    64
```

A.4 Matlab functions

Matlab includes many standard functions. Each function is a block of code that accomplishes a specific task. Commonly used constants such as "pi" (π), and "i" or "j" for the square root of -1, are also incorporated into Matlab. e (the base of natural logarithm) is not included, to obtain e one should use $\exp(1)$ (exponent of 1).

Table A.1: Trigonometric and elementary math functions

sin(x)	Sine of the elements of x
cos(x)	Cosine of the elements of x
asin(x)	Arcsine of the elements of x
acos(x)	Arccosine of the elements of x
tan(x)	Tangent of the elements of x
atan(x)	Arctangent of the elements of x
abs(x)	Absolute value of the elements of x
sqrt(x)	Square root of x
imag(x)	Imaginary part of x
real(x)	Real part of x
conj(x)	Complex conjugate of x
log(x)	Natural logarithm of the elements of x
log10(x)	10 based logarithm of the elements of x
exp(x)	Exponential of the elements of x
sign(x)	Sign of x

Matlab has available most trigonometric and elementary math functions as shown in Table A.1.

Some functions for matrix properties and manipulation are given in Table A.2. To see how a function should be used, type *help function name* at the Matlab command window.

Table A.2: Matrix manipulation

inv(x)	Inverse of a matrix x
eig(x)	Eigenvalues of the matrix x
det(x)	Determinant of matrix x
rank(x)	Rank of matrix x
eye, ones, zeros, diag	Matrix building functions

A.5 Polynomials

A polynomial is represented by a vector. To create a polynomial in Matlab, the coefficients of the polynomial should be entered in descending order. For instance, to enter the following polynomial:

$$p(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$$

enter it as a vector in the following manner:

```
>> p = [1 3 -15 -2 9]
```

Matlab can interpret a vector of length $n+1$ as an nth order polynomial. Thus, also zero coefficients must be entered in the proper places. For instance

$$p(s) = s^4 + 1$$

would be represented in Matlab as:

```
>> p = [1 0 0 0 1];
```

Some functions to be used for polynomials are given in Table A.3:

Table A.3: Functions for polynomials

roots(p)	Roots of polynomial p
polyval(p,value)	Value of polynomial p at value
conv(p,q)	Polynomial multiplication
deconv(p,q)	Divide two polynomials

A.6 Loops and logical statements

Matlab provides loops and logical statements for programming, like *for*, *while*, and *if* statements. The general forms are:
for variable = expression, statement, ..., statement *end*;
while variable, statement, ..., statement, *end*
if variable, statements, *end*

A.7 Plotting

The simple *plot(x,y)* function will plot the vector y versus the vector x.

Let us plot a sine wave as a function of time. First define the time vector and then compute the $\sin$ value at each time index.

```
>> t=0:0.25:7; y = sin(t); plot(t,y)
```

The result is shown in Figure A.1.

Basic plotting is very easy in Matlab, and the plot command has extensive add-on capabilities. These capabilities include many functions such as those presented in Table A.5.

Let us now see 3D plotting. Consider the function $f : \mathbb{R} \to \mathbb{R}$, $f(x, y) = x^2 + y^2$. To graphically represent this function, first the values of x and y have to be defined using *meshgrid*.

Table A.4: Plotting

plot(x,y)	Plots vector y versus vector x.
semilogx(x,y)	Plots vector y versus vector x. The y-axis is log10, the x-axis is linear.
semilogy(x,y)	Plots vector y versus vector x. The x-axis is log10, the y-axis is linear.
loglogx,y)	Plots vector x versus vector y. Both axes are logarithmic.
mesh(x,y,z)	Creates a 3-D mesh surface.
contour(x,y,z)	Plots the countour of a function (2D).

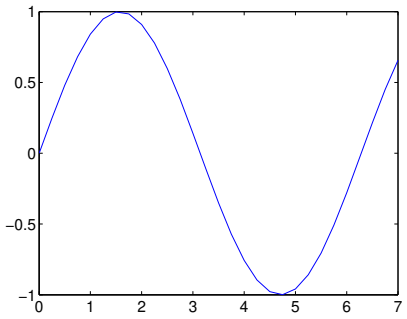


Figure A.1: Plot of $\sin(t)$.

```
>> [x,y]=meshgrid(-1:.01:1);
```

Different grids can also be defined for x and y, as follows:

```
>> [x,y]=meshgrid(-1:.01:1,-2:.01:2);
```

Then, the graphic (see Figure A.2) is obtained by using *mesh*:

```
>> mesh(x,y,x.^2+y.^2)
```

This in effect means that the value of the function is computed for each point on the grid generated by "meshgrid". In many cases more conclusions can be drawn if, instead of the 3D representation, one inspects the projection of the representation on the x-y plane, specifically the *contour* plot, see Figure A.3:

```
>> contour(x,y,x.^2+y.^2)
```

Table A.5: Plotting accessories

grid	Toggles a grid on and off in the current figure.
axis	Controls axis scaling and appearance
title('text')	Adds 'text' at the top of the current axis
xlabel('text')	Labels the x-axis with 'text'
ylabel('text')	Labels the y-axis with 'text'
subplot	Creates axes in tiled positions

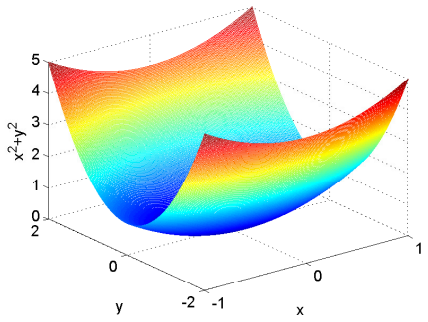


Figure A.2: Graphical representation of $x^2 + y^2$.

A.8 Toolboxes and m-files

The functions in Matlab are in general grouped in toolboxes. For instance, the Control systems Toolbox contains functions of direct use in control engineering. It provides commands for Bode plots, time responses, control design and so on. There are many other toolboxes available for MATLAB, e.g: Optimization, Symbolic Math, Identification, Image Processing, Neural Networks, Spline Functions, Robust Control, Adaptive Control, etc.

The toolboxes are actually written in MATLAB (that is, they use the statements and commands of the MATLAB language). They consist of collections of files, called m-files (since they have the filename extension .m). An m-file is an ASCII file created using any text editor, and containing a sequence of MATLAB commands, typed exactly as they would be from the keyboard when using MATLAB.

Example. Create an m-file called *garbage.m* containing nothing but the following lines:

```
a=[1 2 3; 2 84; 1 7 9];
inv(a)
eig(a)
```

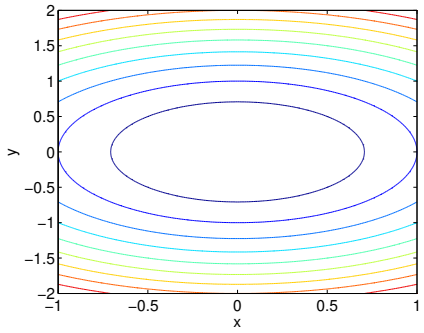


Figure A.3: Contour plot of $x^2 + y^2$.

and simply enter the filename (without the .m extension) in response to the MATLAB prompt. This will execute the commands in the file.

```
>> garbage
```

would have exactly the same result as entering the original commands. The file garbage.m has effectively become a new MATLAB command. Such files are called script files.

Another type of m-file is a *function* file. In contrast with the script files the function files have a name following the word "function" at the beginning of the file. The filename has to be the same as the function name, and it must not start with numbers or contain mathematical operations. The function statement syntax is:

function [output_arguments] = function_name(input_arguments)

The input arguments are variables passed to the function. The output arguments are returned.

Example. Create a m-file called *myfunc.m* which contains the following lines:

```
function [sum, product] = myfunc(x,y)
    sum = x+y;
    product = x*y;
```

The function will return the sum and product of two numbers and it can be called in the following way:

```
>> a=10;
>> b=25.9;
>> [alpha,beta]=myfunc(a,b)
or simply
>> [alpha,beta]=myfunc(10, 25.9)
```

The variable alpha will have the value of the sum of a and b and beta the value of the product.

A.9 Symbolic math

The Symbolic Math toolbox allows one to work with symbolic variables. A symbolic variable x is defined as

```
>> syms x
```

Standard Matlab operations and functions can be used on symbolic variables, and the returned result will be symbolic, i.e., generic variables that can be used without values.

```
>> syms x y z
>> f=x+y
f =
x+y
>> g=2*y+z
g =
2*y+z
>> h=f*g
h =
(x+y)*(2*y+z)
>> f=[x y^2 z]
f =
[   x, y^2,   z]
```

To evaluate a symbolic variable at a given value, one can either use *subs* (for one variable)

```
subs(f,x,1)
ans =
[   1, y^2,   z]
```

or *eval*, after specifying the values

```
>> x=1; y=2; z=3;
>> eval(f)
ans =
     1     4     3
```

Symbolic functions can be differentiated or integrated, using *diff* or *int*. In general, symbolic operations must be performed **before** one evaluates the variables.

The symbolic function h defined above can be differentiated wrt. one variable as

```
diff(h,x)
ans =
2*y+z
diff(diff(h,x),y)
ans =
2
```

or wrt. all the variables as

```
g=jacobian(h)
g =
[     2*y+z, 4*y+z+2*x,        x+y]
```

the result being the vector of partial derivatives. Differentiation of a vector function wrt. all the variables results in a symbolic matrix (function):

```
jacobian(g)
ans =
[ 0, 2, 1]
[ 2, 4, 1]
[ 1, 1, 0]
```

One can also *solve* symbolic equations

```
>> syms x
>> sol=solve('x^2+3*x=2')
sol =
 -3/2+1/2*17^(1/2)
 -3/2-1/2*17^(1/2)
```

or systems of equations

```
>> syms x y
>> sol=solve('x^2+3*x=2','x+y=3')
sol =
    x: [2x1 sym]
    y: [2x1 sym]
>> sol.x
ans =
 -3/2+1/2*17^(1/2)
 -3/2-1/2*17^(1/2)
>> sol.y
ans =
 9/2-1/2*17^(1/2)
 9/2+1/2*17^(1/2)
```

The results are treated as symbolic variables. To obtain the exact value, they must be evaluated:

```
>> eval(sol.x)
ans =
    0.5616
   -3.5616
>> eval(sol.y)
ans =
    2.4384
    6.5616
```

Bibliography

Afshari, S., Aminshahidy, B., and Pishvaie, M. R. (2011). Application of an improved harmony search algorithm in well placement optimization using streamline simulation. *Journal of Petroleum Science and Engineering*, 78:664–678.

Amarantini, D., Rao, G., and Berton, E. (2010). A two-step EMG-and-optimization process to estimate muscle force during dynamic movement. *Journal of Biomechanics*, 43:1827–1830.

Arulampalam, S., Maskell, S., Gordon, N. J., and Clapp, T. (2002). A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Beck, A. T. and de Santana Gomes, W. J. (2012). A comparison of deterministic, reliability-based and risk-based structural optimization under uncertainty. *Probabilistic Engineering Mechanics*, 28:18–29.

Eykhoff, P. (1974). *System identification – state and parameter estimation.* John Wiley & Sons.

Hancock, H. (1960). *Theory of maxima and minima.* Dover, New York.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(1):35–45.

Khalil, H. K. (2002). *Nonlinear Systems.* Prentice-Hall, Upper Saddle River, NJ, USA.

Kitayama, S., Arakawa, M., and Yamazaki, K. (2011). Differential evolution as the global optimization technique and its application to structural optimization. *Applied Soft Computing*, 11:3792–3803.

Luenberger, D. G. (1966). Observers for multivariable systems. *IEEE Transactions on Automatic Control*, 11(2):190–197.

Narendra, K. S. and Annaswamy, A. M. (1989). *Stable Adaptive Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Perez, R. and Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers and Structures*, 85:1579–1588.

Raica, P. (2009). *Optimization*. UT Press, Cluj-Napoca.

Rao, R., Savsani, V., and Vakharia, D. (2011). Teaching learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43:303–315.

Rao, S. S. (1978). *Optimization – theory and applications*. Wiley Eastern Limited.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Application*. Artech House.

Glossary

Conventions and notations

The following conventions are used:

- Capital letters denote matrices, lowercase bold letters denote vectors.
- All the vectors used (except for the derivative) are column vectors. The transpose of a vector is denoted by the superscript T. For instance, the transpose of $\boldsymbol{x}$ is $\boldsymbol{x}^T$.

Notations:

I	identity matrix
0	zero matrix
$A > 0$	A is positive definite (matrix)
$a \succeq 0$	each entry of the vector a is non-negative
$\widehat{s}$	estimated value
$\|\| \cdot \|\|$	norm of a vector/ induced norm of a matrix
$\frac{\partial f}{\partial \boldsymbol{x}}$	partial derivatives of the function f
H	Hessian (matrix of second-order derivatives) of a function