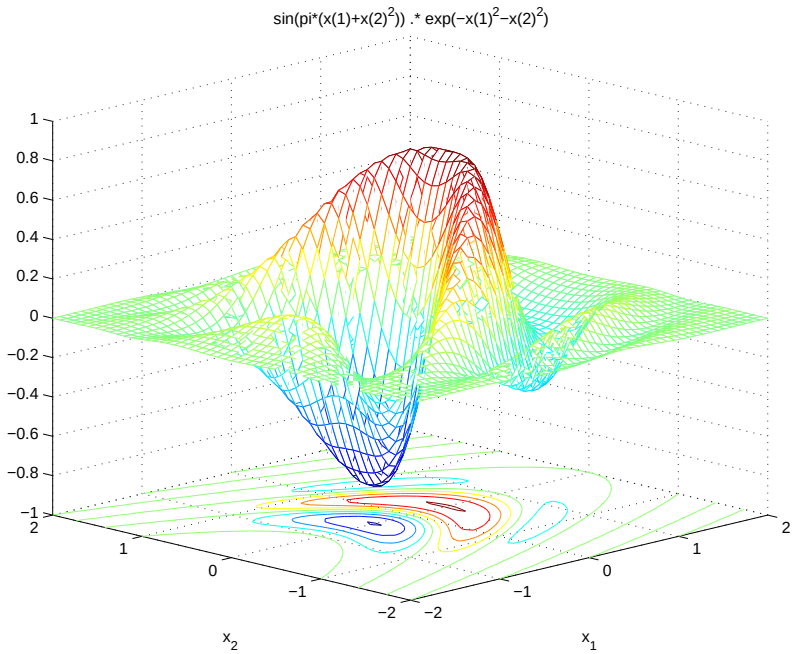
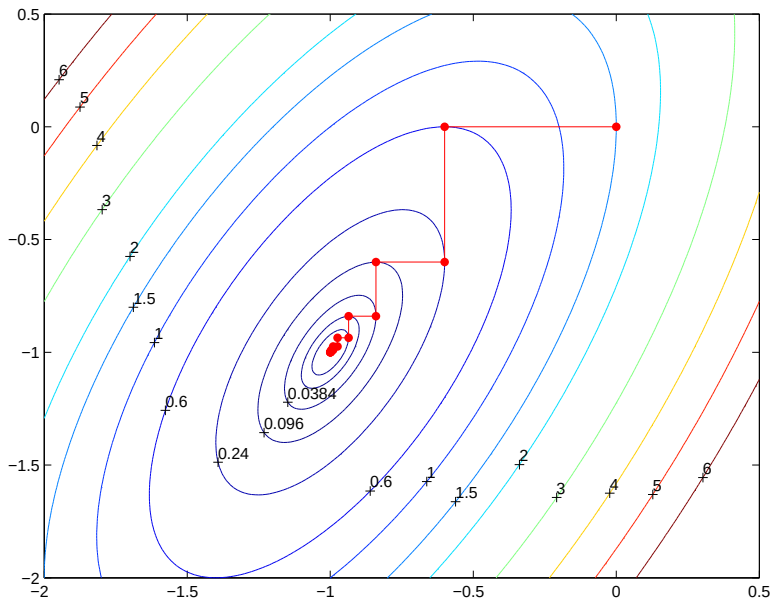
Optimization Methods

R. S. Womersley





School of Mathematics, University of New South Wales

©R. S. Womersley 1996 – 2003

Contents

Notation **ix**

1 Optimization Problems **1**

1.1 Introduction . . . 1
1.1.1 Mathematical models of physical problems . . . 2
1.1.2 Optimization problems . . . 3
1.1.3 Standard form of an optimization problem . . . 4
1.1.4 Exercises . . . 10
1.2 Mathematical Background . . . 12
1.2.1 Definitions of local and global extrema . . . 12
1.2.2 Existence of extrema . . . 14
1.2.3 Gradients and Hessians . . . 15
1.2.4 Vectors and lines in $\mathbb{R}^n$ . . . 17
1.2.5 Taylor series . . . 19
1.2.6 Positive definite matrices . . . 19
1.2.7 Vector norms . . . 21
1.2.8 Logic . . . 26
1.2.9 Exercises . . . 27
1.3 Problem Structure . . . 29
1.3.1 Number of variables n . . . 29
1.3.2 Objective function $f(\mathbf{x})$ . . . 29
1.3.3 Specifying the feasible region Ω . . . 31
1.3.4 Available information . . . 34
1.3.5 Exercises . . . 37
1.4 Practical Issues . . . 37
1.4.1 Rounding errors . . . 38
1.4.2 Testing for zero . . . 38
1.4.3 Differentiability . . . 39
1.4.4 Calculating derivatives . . . 40
1.4.5 Finite difference approximations . . . 40
1.4.6 Computational complexity . . . 41
1.4.7 Efficient implementations . . . 44
1.4.8 Desired solution . . . 44
1.4.9 Exercises . . . 44
1.5 Algorithms . . . 45
1.5.1 Global convergence . . . 45
1.5.2 Local convergence . . . 45
1.5.3 Convergence conditions . . . 47
1.5.4 Affine invariance . . . 48
1.5.5 Comparing methods . . . 48
1.5.6 Exercises . . . 49

2 Convexity **51**
2.1 Convex sets . . . 51
2.1.1 Exercises . . . 54
2.2 Convex Functions . . . 54
2.2.1 Exercises . . . 58
2.3 Convex programming problems . . . 60
2.3.1 Exercises . . . 61

3 Optimality Conditions **63**
3.1 Unconstrained Problems . . . 63
3.1.1 First order necessary conditions . . . 63
3.1.2 Second order conditions . . . 64
3.1.3 Exercises . . . 69
3.2 Equality Constraints . . . 71
3.2.1 First order necessary conditions . . . 73
3.2.2 Second order conditions . . . 76
3.2.3 Sensitivity analysis . . . 80
3.2.4 Exercises . . . 82
3.3 Inequality Constraints . . . 83
3.3.1 First order conditions . . . 85
3.3.2 Second order conditions . . . 89
3.3.3 Sensitivity analysis . . . 92
3.3.4 Exercises . . . 95

4 Methods for One-dimensional Problems **97**
4.1 Fixed point iteration . . . 98
4.2 Newton's Method . . . 98
4.3 Secant Method . . . 99
4.4 Polynomial interpolation . . . 100
4.4.1 Cubic Interpolation . . . 100
4.4.2 Quadratic Interpolation . . . 101
4.5 Bracketing Methods . . . 101
4.5.1 Bisection . . . 102
4.5.2 Golden Section Search . . . 102
4.6 Nonsmooth Problems . . . 102
4.7 Exercises . . . 103

5 Methods for Unconstrained Problems **107**
5.1 Introduction . . . 107
5.1.1 Quadratic models . . . 107
5.1.2 Line search methods . . . 108
5.1.3 Approximate line searches . . . 109
5.1.4 Trust region methods . . . 111
5.1.5 Convergence conditions . . . 113
5.1.6 Desirable properties . . . 114
5.1.7 Exercises . . . 114
5.2 Steepest descent . . . 115
5.2.1 Exercises . . . 119
5.3 Newton's method . . . 120
5.3.1 Modified Newton's method . . . 122
5.3.2 Exercises . . . 123
5.4 Quasi-Newton methods . . . 125
5.4.1 Updating Hessian approximations . . . 126
5.4.2 Properties . . . 127
5.4.3 Exercises . . . 129

5.5 Sums of Squares and Nonlinear Equations . . . 130
5.5.1 Nonlinear equations . . . 130
5.5.2 Gauss-Newton method . . . 132
5.5.3 Well-determined systems . . . 134
5.5.4 Exercises . . . 134
5.6 Methods using only function values . . . 140
5.6.1 Conjugate direction methods . . . 141
5.6.2 Nelder-Meade Simplex method . . . 142
5.7 Large scale methods . . . 142
5.7.1 Conjugate gradient methods . . . 142
5.7.2 Newton methods using sparse matrix factorizations . . . 146
5.7.3 Incomplete Newton methods . . . 147
5.7.4 Sparse quasi-Newton approximations . . . 147
5.7.5 Limited memory quasi-Newton methods . . . 147
5.7.6 Exercises . . . 147

6 Methods for Linearly Constrained Problems 149
6.1 Introduction . . . 149
6.1.1 Finding a feasible point . . . 150
6.1.2 Feasible directions . . . 151
6.1.3 Feasible line searches . . . 152
6.1.4 Active set methods . . . 152
6.1.5 Equality constrained subproblems . . . 153
6.1.6 Exercises . . . 154
6.2 An active set method for linear programming . . . 155
6.2.1 Exercises . . . 157
6.3 Active set method for quadratic programming . . . 160
6.3.1 Exercises . . . 163
6.4 Interior Point Methods . . . 164

7 Methods for Nonlinearly Constrained Problems 165
7.1 Penalty and Merit Functions . . . 165
7.1.1 Sum of squares penalty function . . . 168
7.1.2 Augmented Lagrangian penalty function . . . 169
7.1.3 Exact penalty functions . . . 170
7.1.4 Exercises . . . 170
7.2 Sequential quadratic programming . . . 170
7.2.1 Sequential linear programming . . . 170
7.2.2 Sequential quadratic programming . . . 172
7.2.3 Exercises . . . 174

8 Portfolio Optimization [H] 175
8.1 Introduction . . . 175
8.1.1 Asset Portfolios . . . 177
8.2 Mean-variance models . . . 179
8.2.1 Maximizing Returns . . . 179
8.2.2 Minimizing Risk . . . 181
8.2.3 Balancing Objectives . . . 181
8.3 Scenarios . . . 182
8.4 Other measures of risk . . . 183
8.4.1 A Semi-variance Model . . . 184
8.4.2 Skewness Model . . . 185
8.5 Exercises . . . 186
8.6 References . . . 187

9 Approximation Problems [H] 189
9.1 Function Approximation . . . 189
9.1.1 Introduction . . . 189
9.1.2 Inner products . . . 192
9.1.3 Linear least squares . . . 192
9.1.4 Basis functions . . . 193
9.1.5 Orthogonal functions . . . 194
9.1.6 Exercises . . . 195
9.2 Data Fitting . . . 195
9.2.1 Linear Least Squares . . . 196
9.2.2 Linear ℓ_1 . . . 197
9.2.3 Linear ℓ_∞ . . . 197
9.2.4 Nonlinear Least Squares . . . 198
9.2.5 Nonlinear ℓ_1 . . . 199
9.2.6 Nonlinear ℓ_∞ and Minimax . . . 199
9.2.7 Exercises . . . 199
9.2.8 References . . . 201

10 Nonsmooth Optimization [H] 203
10.1 Introduction . . . 203
10.2 Convex optimization . . . 203
10.2.1 Derivatives of convex functions . . . 203
10.2.2 Directional Derivatives . . . 204
10.2.3 Subgradients and Subdifferentials . . . 204
10.2.4 Optimality Conditions . . . 208
10.2.5 Exercises . . . 209
10.3 Convex Composite Optimization . . . 209
10.3.1 Convex composite functions . . . 209
10.3.2 Generalized gradients . . . 210
10.3.3 Optimality conditions . . . 210
10.3.4 Numerical Methods . . . 212
10.3.5 Linear ℓ_1 . . . 215
10.3.6 Exercises . . . 216
10.4 References . . . 217

11 Probabilistic Methods and Global Optimization [H] 219
11.1 Quasi Monte-Carlo methods . . . 219
11.2 Simulated Annealing . . . 219
11.3 Genetic Algorithms . . . 219

12 Stochastic Optimization [H] 221
12.1 Introduction . . . 221

A Matrix Algebra 223
A.1 Vectors and Matrices . . . 223
A.2 Special Matrices . . . 226
A.3 Inverses and Determinants . . . 229
A.4 Subspaces, Dimension and Rank . . . 230
A.4.1 Range and Null Spaces . . . 230
A.4.2 Orthogonality and Conjugacy . . . 231
A.4.3 Rank-1 Matrices . . . 231
A.5 Eigenvalues and Eigenvectors . . . 232
A.5.1 Eigenvalues of Special Matrices . . . 232
A.6 Vector and Matrix Norms . . . 233
A.6.1 Vector Norms . . . 233

A.6.2 Matrix Norms . . . 235
A.7 Matrix Factorizations . . . 236
A.7.1 LU Factorization . . . 237
A.7.2 Cholesky Factorization . . . 237
A.7.3 QR Factorization . . . 238
A.7.4 Singular Value Decomposition (SVD) . . . 239
A.7.5 Sparse factorizations . . . 240
A.8 Systems of Linear Equations . . . 241
A.8.1 Triangular systems . . . 241
A.8.2 Well-determined Systems . . . 241
A.8.3 Numerical Sensitivity . . . 242
A.8.4 Over-determined Linear Systems . . . 243
A.8.5 Under-determined Linear Systems . . . 245
A.9 Implementations . . . 246
A.9.1 Floating point operations . . . 247
A.9.2 Memory access . . . 247
A.9.3 Vectorization . . . 249
A.10 References . . . 250

B Basic Statistics **253**
B.1 Random variables . . . 253
B.1.1 Probabilities . . . 253
B.1.2 Mean and Variance . . . 254
B.1.3 Normal and Lognormal distributions . . . 255
B.1.4 Correlation . . . 256
B.2 Random vectors . . . 258
B.2.1 Multivariate normal distribution . . . 259

Bibliography **261**

Index **269**

Notation

Usually lower case letters represent vectors or functions, capital letters represent matrices, and Greek lower case letters represent scalars.

n = number of variables;

$\mathbb{R}^n$ = space of all real n-dimensional vectors;

Ω = feasible region = set of points in $\mathbb{R}^n$ satisfying ALL the constraints;

$\mathbf{x}$ = vector of variables, $\mathbf{x} \in \mathbb{R}^n$;

$\mathbf{x}^*$ = a solution point or a stationary point;

$\mathbf{x}^{(k)}$ = k th iterate generated by a numerical method;

$f(\mathbf{x})$ = objective function;

$\mathbf{g}(\mathbf{x}) \equiv \nabla f(\mathbf{x})$ = gradient of the objective function $f(\mathbf{x})$, $\nabla f(\mathbf{x}) \in \mathbb{R}^n$;

$G(\mathbf{x}) \equiv \nabla^2 f(\mathbf{x})$ = Hessian of the objective function $f(\mathbf{x})$, $\nabla^2 f(\mathbf{x}) \in \mathbb{R}^{n\times n}$;

$c_i(\mathbf{x})$ = i th constraint function;

$\mathcal{E}$ = set of indices of the Equality constraints: $c_i(\mathbf{x}) = 0$ for $i \in \mathcal{E}$;

$\mathcal{I}$ = set of indices of the Inequality constraints: $c_i(\mathbf{x}) \leq 0$ for $i \in \mathcal{I}$;

$m_{\mathcal{E}}$ = number of equality constraints; $m_{\mathcal{E}} = |\mathcal{E}|$;

$m_{\mathcal{I}}$ = number of inequality constraints; $m_{\mathcal{I}} = |\mathcal{I}|$;

m = total number of constraints; $m = m_{\mathcal{E}} + m_{\mathcal{I}} = |\mathcal{E}| + |\mathcal{I}|$;

$\mathbf{a}_i(\mathbf{x}) \equiv \nabla c_i(\mathbf{x})$ = gradient of the i th constraint, $\nabla c_i(\mathbf{x}) \in \mathbb{R}^n$;

$\nabla^2 c_i(\mathbf{x})$ = Hessian of the i th constraint, $\nabla c_i(\mathbf{x}) \in \mathbb{R}^{n\times n}$;

λ_i = Lagrange multiplier corresponding to constraint $c_i(\mathbf{x})$;

$\mathcal{L}(\mathbf{x}, \lambda) \equiv f(\mathbf{x}) + \sum_{i\in\mathcal{E}\cup\mathcal{I}} \lambda_i c_i(\mathbf{x})$ = Lagrangian function;

$W(\mathbf{x}) \equiv \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla^2 f(\mathbf{x}) + \sum_{i\in\mathcal{E}\cup\mathcal{I}} \lambda_i \nabla^2 c_i(\mathbf{x})$ = Hessian of the Lagrangian function;

$\mathcal{A}(\mathbf{x}) \equiv \{i \in \mathcal{E} \cup \mathcal{I} : c_i(\mathbf{x}) = 0\}$ = set of indices of the active constraints;

$t(\mathbf{x}) = |\mathcal{A}(\mathbf{x})|$ = number of active constraints at $\mathbf{x}$;

$A(\mathbf{x}) \equiv [\mathbf{a}_i(\mathbf{x}) : i \in \mathcal{A}(\mathbf{x})]$ = matrix of gradients of active constraints, $A(\mathbf{x}) \in \mathbb{R}^{n\times t(\mathbf{x})}$;

$Z(\mathbf{x})$ = basis for null space of $A(\mathbf{x})^T$, so $Z(\mathbf{x}) \in \mathbb{R}^{n\times(n-t(\mathbf{x}))}$ has full rank and $A(\mathbf{x})^T Z(\mathbf{x}) = \mathbf{0}$;

$\mathbf{g}_Z(\mathbf{x}) \equiv Z(\mathbf{x})^T \nabla f(\mathbf{x})$ = reduced gradient;

$G_Z(\mathbf{x}) \equiv Z(\mathbf{x})^T \nabla^2 f(\mathbf{x}) Z(\mathbf{x})$ = reduced Hessian;

$\mathbf{d}^{(k)}$ = search direction on the k th iteration;

$\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}$ = line ($\alpha \in \mathbb{R}$) passing through $\mathbf{x}^{(k)}$ in the direction $\mathbf{d}^{(k)}$;

$\alpha^{(k)}$ = step length on the k th iteration;

$\ell_k(\alpha) \equiv f\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right)$ = function value along the line;

$\mathbf{s}^{(k)} \equiv \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \alpha^{(k)} \mathbf{d}^{(k)}$ = difference in successive iterates;

$\mathbf{y}^{(k)} \equiv \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$ = difference in successive gradients;

$B^{(k)}$ = quasi-Newton approximation to a Hessian matrix, $B^{(k)} \in \mathbb{R}^{n\times n}$;

$\mathbf{r}^{(k)}$ = vector of residuals $\mathbf{r}(\mathbf{x})$ evaluated at $\mathbf{x}^{(k)}$, $\mathbf{r}(\mathbf{x}) \in \mathbb{R}^m$;

$J^{(k)}$ = Jacobian matrix $J(\mathbf{x})$ evaluated at $\mathbf{x}^{(k)}$, $J(\mathbf{x}) \in \mathbb{R}^{m\times n}$;

Chapter 1

Optimization Problems

1.1 Introduction

Optimization problems deal with finding the "best" way to do a task, subject to any constraints or restrictions on the actions that may be taken to accomplish the task. Many everyday problems can be posed as optimization problems, with applications arising in many disciplines, including commerce and economics (portfolio optimization), management (scheduling), engineering (structural design) and science. Solving an optimization problem requires the formulation of a mathematical model of the physical problem, the analysis and solution (either by analytic or numerical techniques) of the mathematical model, and the interpretation of the solution of the mathematical model in terms of the original physical problem.

Optimization methods are part of wider areas known as *Operations Research*, *Management Science* and *Mathematical Programming* (see [GH96, NRKT89, Tah92, HL90] for example). This course concentrates on continuous nonlinear optimization problems, as distinct from linear programming problems (see [Chv83, Mur81, Sai95, Sch98] for example) and discrete or combinatorial problems (see [LLKS85, NW88, PS82] for example).

Optimization has its foundations in the development of calculus by Newton and Leibniz in the 17th century. Now the idea of setting the derivative $f'(x) = 0$ to find a minimizer or maximizer of a continuously differentiable function of a single variable is seen at an early stage. The work of the Lagrange in the 18th century on equality constrained problems is also covered in most multi-variable calculus courses. Numerical optimization, and in particular the solution of large optimization models using computer programs, started with the work of Dantzig [Dan59, Dan91] in the late 1940s and 1950s on the simplex method for linear programming. The size of problems that can be solve has since tracked the rapidly growing power of computers and the increasingly sophisticated algorithms and software that has become available. A key aspect of this has been the development of stable efficient matrix algebra software (for example LAPACK [ABB+95]), and in particular the advances in sparse matrix technology. In 1984 the work of Karmarkar [Kar84] on a polynomial time algorithm for linear programming was another milestone in the history of optimization methods. Karmarkar's work has lead to the development of interior point methods (see [Wri97]) for many optimization problems, allowing yet larger problems to be solved. Nonlinear problems with a few hundred variables can now be solved relatively routinely, while linear programs of up to a million variables have been solved by state of the art codes on parallel computers. Some of the history of mathematical programming is provided by [LRS91, Nas90].

Nocedal and Wright [NW99], Fletcher [Fle87] and Dennis and Schnabel [DS83] provide good mathematical treatments of nonlinear optimization methods. Gill, Murray and Wright [GMW81] provide a more practical numerical approach, while More and Wright [MW93] provides both a summary of the methods and the software available. Polak [Pol97] concentrates on methods that are relevant to engineering optimization problems. Basic methods with source code are in both the C and Fortran versions of Numerical Recipes [PTVF92].

There are modelling languages which make the formulation of optimization problems easier and which can use a variety of packages for the numerical solution of the problem. Examples are GAMS [BKM88, BM82] and AMPL [FGK93].

These notes aim to provide an understanding of the theoretical basis of optimization methods and the techniques used to develop numerical methods for solving them. The aim is not to provide a complete coverage of the theory of optimization methods and every possible solution algorithm. Only an elementary knowledge of multi-variable calculus and linear algebra is assumed. Section 1.2 summarizes some of the multi-variable calculus that is required, while Appendix A provides a summary of relevant parts of matrix algebra. Appendix B provides an elementary summary of statistical concepts of particular relevance to optimization.

The packages `Maple` [Cha92, Cha91] and MATLAB [Mat01], in particular the MATLAB optimization toolbox, are used in the solution of many of the examples and exercises. `Maple`, with its primary focus on symbolic calculations, is particularly useful in calculating derivatives, checking optimality conditions and situations where exact arithmetic is required. The knowledge of `Maple` from the first year computing notes is useful. On the other hand, MATLAB, which has a matrix as its basic object and is numerically oriented, is particularly useful in the development and testing of numerical methods. Both `Maple` and MATLAB are used to visualize multi-variable functions and the progress of numerical methods. Useful references on `Maple` include [Red93, Cor95, Joh93] while references on MATLAB include [HH00, Pra01, GH97]. Appendix A on Matrix Algebra has examples of some of the MATLAB commands. `Maple` is available though the symbolic mathematics toolbox in MATLAB, but will be used as a separate package in these notes, as it was used for first and second year subjects.

A number of World Wide Web (WWW) resources on optimization are also available. Links to some of these are available through the WWW page

`http://www.maths.unsw.edu.au/~rsw/optlink.html`

Of particular interest is the *Optimization Technology Centre* and links to software providers such as NAG, Visual Numerics, CPLEX, OSL and many others.

This chapter considers the formulation of optimization problems, what is meant by a solution to an optimization problem, the basic mathematical background required, and practical issues that arise when solving optimization problems.

1.1.1 Mathematical models of physical problems

When modelling any "real life" problem there are many issues that need to be considered. The following points apply to most scientific problems, including optimization problems. The typical steps involved in solving a problem are:

1. Describe the physical problem.
   (a) Is the description consistent?
   (b) Is the description complete?
2. Produce a mathematical model of the physical problem. Aspects to be considered include:
   (a) What approximations or simplifications must be made to produce a tractable mathematical model?
   (b) Is an analytical solution of the mathematical model possible?
      i. What mathematical background is required?
   (c) What analytical properties does the mathematical model have?
   (d) What is required for a numerical solution of the mathematical model?
      i. What computing equipment is available or required?
      ii. What programming language(s) are to be used?
      iii. What mathematical software is available?
      iv. What are the appropriate numerical methods?
   (e) Solve the mathematical model as efficiently and accurately as required.
      i. Interpret and visualize the solution.

ii. Does the numerical solution agree with the analytical properties of the mathematical model?

iii. *Does the solution of the mathematical model make sense in terms of the physical problem*?

3. Revise and re-solve the mathematical model in light of the results obtained so far.

Frequently many cycles through the above steps are required, as interpretation and validation of the results may indicate

- Inconsistencies or inadequacies in the physical description of the problem.
- Inadequacies in the approximations or simplifications made to produce the mathematical model.
- Errors in the numerical solution of the mathematical model.

1.1.2 Optimization problems

An **optimization problem** is to find the "best" way of doing something subject to various conditions or restrictions on what can be done. There are often many possible definitions of "best" and many restrictions on what actions or decisions are allowed. Examples of optimization problems include

- Choosing an investment portfolio to minimize its risk or to maximize its return.
- The design of the shape of an aircraft (or a critical part like the wings) to maximize the aircraft performance or minimize the construction cost.
- Minimizing the amount of water required in the cooling of continuously cast steel.
- Maximizing the economic lifetime of a coal mine.
- Scheduling vehicles to minimize the number of trucks required to deliver a set of items.
- Maximizing the profit obtained from selling different varieties of beers.
- Approximating a function by a low order polynomial.
- Finding a simple function to represent a series of measurements.

The first step is to formulate a mathematical model of the optimization problem, so we can use our mathematical knowledge to solve it. This process can be split into a number of steps.

1. Decide what variables are needed.

   The variables usually provide the answers to the questions posed, but sometimes additional variables are needed in the mathematical model. It is essential that all variables are precisely defined.

2. Determine the constraints or restrictions (if any) on the variables.

3. Find a mathematical representation of "best" in terms of the variables.

These steps are not independent of the solution process, as the mathematical model must be made keeping in mind those types of mathematical optimization problems which can be solved (either by hand or more usually using a computer software package). Also when the results of the mathematical optimization problem are applied to the real-life problem extra constraints are often discovered, forcing a revision of the mathematical model. *It is dangerous to accept without careful consideration the results produced by the mathematical model or the software package.* The results must always be checked to see if they make sense in the context of the physical problem from which they came. The verification and maintenance of large mathematical programming models is a significant issue [PKS96].

Modelling languages such as AMPL [FGK93] (A Mathematical Programming Language) and GAMS (General Algebraic Modelling System) [BM82, BKM88] greatly simplify the development of optimization models. The optimization problems produced can be solved by a variety of methods.

These notes are primarily concerned with the mathematics that is used to solve optimization problems. It draws heavily on multi-variable calculus (as you rarely have only a single variable) and matrix algebra. Most practical optimization problems are far too large to be solved by hand, so numerical methods implemented on a computer are required. The efficiency of these numerical optimization methods is important as, despite the advances in computer technology, many optimization problems are so large and complicated that they require significant amounts of computing time.

1.1.3 Standard form of an optimization problem

There are several different ways of posing a mathematical optimization problem. However it is easiest to work with one standard formulation, and to convert problems into that form. The *standard form* of an optimization problem considered in these notes is

$$\begin{array}{ll} Minimize & f(\mathbf{x}) \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & \mathbf{x} \in \Omega. \end{array} \tag{1.1.1}$$

The *variables* (often called decision or policy variables) are $x_1, x_2, \ldots x_n$, where n is the number of variables. Note that $\mathbf{x}$ is a column vector with n elements, while $\mathbf{x}^T$ is a row vector with n elements, so

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^T.$$

The Euclidean length $\|\mathbf{x}\|$ of a vector $\mathbf{x} \in \mathbb{R}^n$ is $\|\mathbf{x}\| = \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$.

The function $f(\mathbf{x})$ is called the *objective function* and is a mathematical representation of "best" in terms of the variables $\mathbf{x}$.

The set Ω is a subset of n-dimensional Euclidean space $\mathbb{R}^n$ called the *feasible region.* A point $\mathbf{x} \in \Omega$ is called a *feasible point.* The set Ω consists of all the points $\mathbf{x} \in \mathbb{R}^n$ which satisfy ALL the constraints. If $\Omega = \mathbb{R}^n$ (that is $-\infty < x_i < \infty$ for $i = 1, \ldots, n$) then the problem is *unconstrained.* If Ω is a strict subset of $\mathbb{R}^n$ then the problem is *constrained.*

The standard formulation is a *minimization* problem. Any problem posed as a maximization problem can be converted into a minimization problem simply by changing the sign of the objective function. The problems

$$\begin{array}{ll} Maximize & \bar{f}(\mathbf{x}) \\ Subject\ to & \mathbf{x} \in \Omega \end{array} \quad \text{and} \quad \begin{array}{ll} Minimize & f(\mathbf{x}) \\ Subject\ to & \mathbf{x} \in \Omega \end{array} \tag{1.1.2}$$

where $f(\mathbf{x}) = -\bar{f}(\mathbf{x})$, have the same solution points $\mathbf{x}^*$, and their optimal objective values are related by $f(\mathbf{x}^*) = -\bar{f}(\mathbf{x}^*)$.

If the constraints are all algebraic in nature (i.e. $2x_1 + x_2^2 = 2$) then you have a *mathematical programming* problem. If the constraints involve differential equations i.e.

$$\frac{dx_1}{dt} = 2x_1 + x_2^2 - 2,$$

then you have an *optimal control* problem. These notes are only concerned with mathematical programming problems.

The feasible region Ω is usually specified by a collection of equations and inequalities in the variables $\mathbf{x}$. The standard specification of the feasible region is

$$\Omega = \{\, \mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \ \forall\ i \in \mathcal{E},\ c_i(\mathbf{x}) \leq 0 \ \forall\ i \in \mathcal{I} \,\}. \tag{1.1.3}$$

Here $\mathcal{E}$ and $\mathcal{I}$ are index sets for the equality and inequality constraints respectively. Note that the right hand sides of these equalities and inequalities are always zero. An inequality constraint $c_i(\mathbf{x}) \geq 0$ is equivalent to $-c_i(\mathbf{x}) \leq 0$, so there is no loss of generality in only considering $\leq$ constraints. The equality

constraint $c_i(\mathbf{x}) = 0$ is equivalent to the two inequality constraints $c_i(\mathbf{x}) \geq 0$ and $c_i(\mathbf{x}) \leq 0$, however this is not used as equality constraints play a key role in determining solution points. Strict inequality constraints $c_i(\mathbf{x}) < 0$ pose additional difficulties, so they are usually approximated by $c_i(\mathbf{x}) + \epsilon \leq 0$ where $\epsilon > 0$ is a small positive constant.

Discrete constraints such as binary variables $x_i = \{0, 1\}$ or integer variables $x_i = \{0, 1, 2, 3, \ldots\}$ are important in modelling yes/no decisions (do you open a new factory) and problems where whole numbers of a quantity are required. (A customer would not be happy with 0.65 of a computer.)

Example 1.1.1 (Post Office Parcel Problem) *At one time the post office regulations were that "the length plus the girth of any parcel sent through the post must not exceed 1.8 metres". The length of a parcel is defined as its longest dimension, and the girth as the distance perpendicular to the length around the parcel. What are the dimensions of the parcel with rectangular sides of largest volume that could then be sent through the post?*

Solution The first step is to carefully define the variables. A sketch of a rectangular parcel in given in Figure 1.1.1. Let

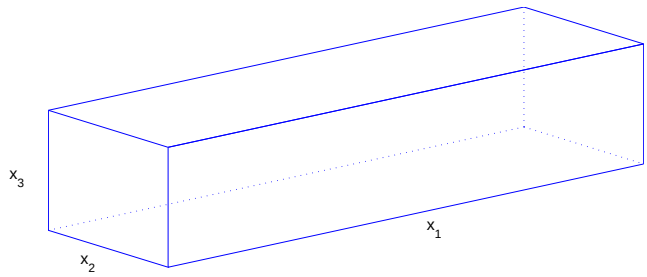


Figure 1.1.1: Variables for post office parcel problem

- x_1 = length of parcel in centimetres,
- x_2 = width of parcel in centimetres,
- x_3 = depth of parcel in centimetres.

The objective is to maximize the volume $= x_1 x_2 x_3$ cubic centimetres. The constraints are

- Length + girth $= x_1 + 2x_2 + 2x_3 \leq 180$. (Note conversion of 1.8m to cm.)
- Nonnegative dimensions: $x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0$. (Ensure physically meaningful variables.)
- Longest dimension x_1: $x_1 \geq x_2,\ x_1 \geq x_3$.

In fact not all these constraints are needed to solve the problem, but it is not immediately obvious which constraints can be removed. The feasible region is sketched in Figure 1.1.2. As the number of variables and the number of constraints increases the feasible region becomes increasingly complex and difficult to visualize.

In standard form the problem is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^3}{Minimize} & f(\mathbf{x}) = -x_1 x_2 x_3 \\ Subject\ to & \mathbf{x} \in \Omega, \end{array}$$

where

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^3 : x_1 + 2x_2 + 2x_3 \leq 180,\ x_1 \geq x_2,\ x_1 \geq x_3,\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \right\}.$$

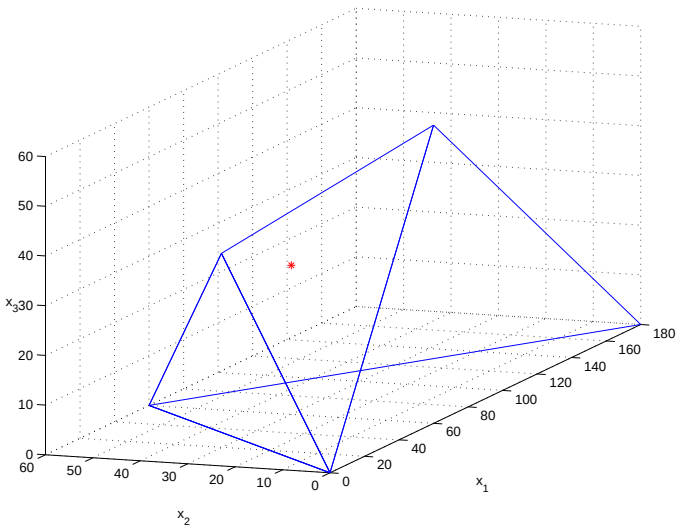


Figure 1.1.2: Feasible region for post office parcel problem

The standard formulation (1.1.3) of the feasible region for the Post Office Parcel problem is

$$\begin{aligned}
c_1(\mathbf{x}) &= x_1 + 2x_2 + 2x_3 - 180 \leq 0, \\
c_2(\mathbf{x}) &= -x_1 + x_2 \leq 0, \\
c_3(\mathbf{x}) &= -x_1 + x_3 \leq 0, \\
c_4(\mathbf{x}) &= -x_1 \leq 0, \\
c_5(\mathbf{x}) &= -x_2 \leq 0, \\
c_6(\mathbf{x}) &= -x_3 \leq 0,
\end{aligned}$$

so $\mathcal{E} = \emptyset$ (the empty set) and $\mathcal{I} = \{1, 2, 3, 4, 5, 6\}$. Note that the order in which the constraints are listed is not important, as long as you are consistent within a problem.

MATLAB scripts to solve the Post Office Parcel problem are in the files `popsol.m`, `popobj.m` available from the course web page.

The post office parcel problem has linear constraints $c_i(\mathbf{x})$, but a nonlinear objective function $f(\mathbf{x})$. An important aspect of optimization problems is to fully exploit the structures of the objective and constraint functions that define the problem, so that an efficient solution method is used. The structure of optimization problems is discussed in more detail in Section 1.3.

Example 1.1.2 (Portfolio optimization) *An investor has assets to invest either in a property trust or a share trust or both. The investor decides to invest at least* 20% *and at most* 80% *of their assets in the property trust, and to invest at least* 10% *of their assets in the share trust. The predicted return on the property trust is* 5% *per annum, and* 12% *per annum for the share trust. Associated with each type of investment there is a risk which the investor decides to model by the historical variance in the returns of the two investments. This also gives the investor an idea of the correlation between returns on the two investments. The covariances in Table 1.1.1, indicate that the property trust has less risk (variance of*

	Property trust	Share trust
Property trust	30%	-20%
Share trust	-20%	45%

Table 1.1.1: Variances in investment returns

30%) than the share trust (variance of 45%*), and that the returns on these two investments tend to move in opposite directions (covariance of* -20%*).*

1. *Define variables that describe the investors choice, and any constraints on these variables.*
2. *Formulate a minimization problem that finds the asset allocation that maximizes the investors predicted return.*
3. *Formulate a minimization problem that finds the asset allocation that minimizes the investors risk.*
4. *Formulate a minimization problem that finds the asset allocation that minimizes a weighted combination of the expected return and the risk.*
5. *Use the* MATLAB *program in the file* `popt.m` *to solve these problems. Try changing the data and see what happens. Describe the solutions to the investor who is not a mathematician.*

Solution

1. As you do not know the amount the investor has available it is easiest to work with fractions or percentages of their available assets. Let

   - x_1 = the fraction of the investor's assets that are allocated to the property trust.
   - x_2 = the fraction of the investor's assets that are allocated to the share trust.

   For this problem there are $n = 2$ variables. Assuming the investor can only invest money they actually have (i.e. they cannot shortsell or borrow money), the definition of the variables implies that
   $$0 \le x_i \le 1 \quad i = 1, 2 \quad \text{and} \quad x_1 + x_2 \le 1.$$
   Note that the constraints $x_i \ge 0$ and $x_1 + x_2 \le 1$ imply that $x_1 \le 1$ and $x_2 \le 1$, so the latter two constraints may not be needed. The investor's decisions to invest at least 20% and at most 80% of their assets in the property trust, and to invest at least 10% of their assets in the share trust, mean that
   $$0.2 \le x_1 \le 0.8, \qquad 0.1 \le x_2 \le 1.$$
   These are simple lower and upper bounds on the variables. The feasible region is thus
   $$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^2 : \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix} \le \mathbf{x} \le \begin{bmatrix} 0.8 \\ 1 \end{bmatrix}, \quad x_1 + x_2 \le 1 \right\}.$$
   All the constraints defining Ω are affine. The feasible region Ω is drawn in Figure 1.1.3

2. The expected return if a fraction x_1 is invested in the property trust and a fraction x_2 is invested in the share trust is
   $$0.05x_1 + 0.12x_2 = \mathbf{r}^T\mathbf{x} \quad \text{where} \quad \mathbf{r} = \begin{bmatrix} 0.05 \\ 0.12 \end{bmatrix}.$$
   Here $\mathbf{r}$ is the vector of predicted returns for the different investments. As the investor wants to maximize their return, this can be posed as
   $$\min_{\mathbf{x} \in \Omega} f_r(\mathbf{x}) \qquad \text{where} \quad f_r(\mathbf{x}) = -\mathbf{r}^T\mathbf{x}.$$
   In this case the objective and all the constraints are linear, so this is a linear programming problem.

3. The risk of the investment portfolio, as measured by the variance of the portfolio, is
   $$0.3x_1^2 - 0.4x_1x_2 + 0.45x_2^2 = \mathbf{x}^T C\mathbf{x}$$
   where
   $$C = \begin{bmatrix} 0.3 & -0.2 \\ -0.2 & 0.45 \end{bmatrix}$$
   is the covariance matrix. Thus the risk associated with the investments is minimized by solving
   $$\min_{\mathbf{x} \in \Omega} f_c(\mathbf{x}) \qquad \text{where} \quad f_c(\mathbf{x}) = \mathbf{x}^T C\mathbf{x}.$$

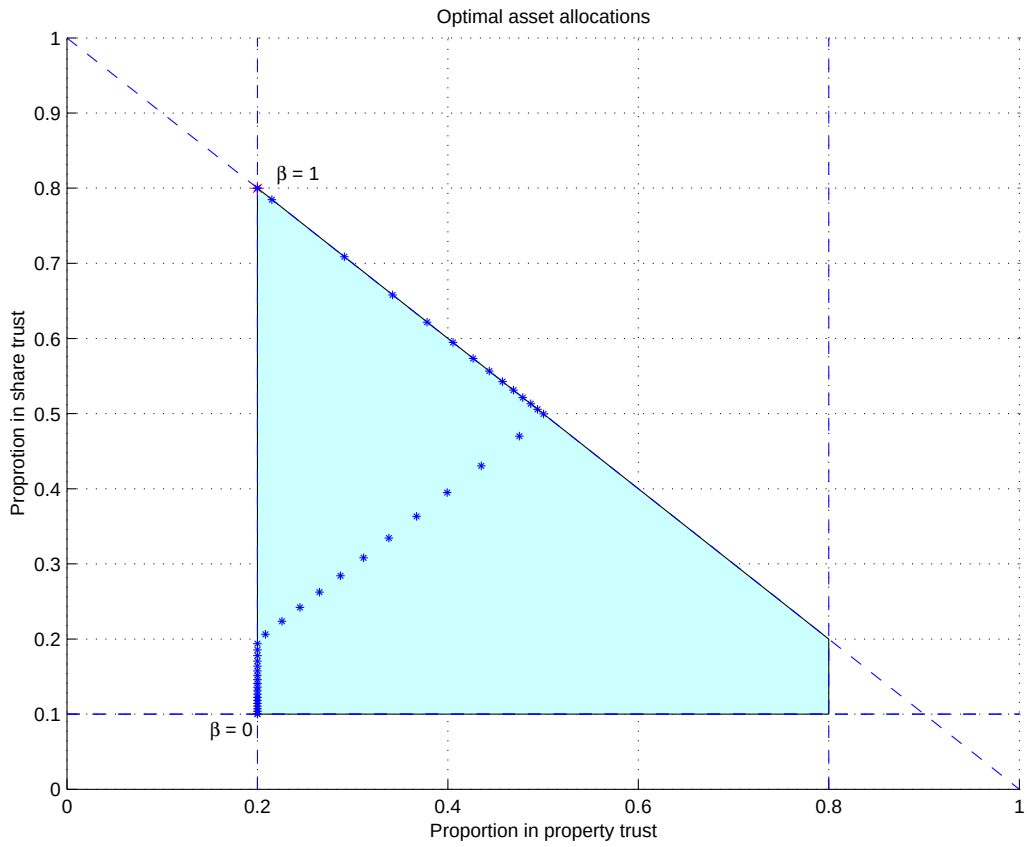


Figure 1.1.3: Feasible region for asset allocations and optimal points

As all the constraints are linear and the objective function is quadratic this is a quadratic programming problem. Moreover a covariance matrix is positive semi-definite, so this is a convex quadratic programming problem. Note that quadratic functions are normally written with a factor of $\frac{1}{2}$ outside the quadratic term, in which case

$$\tfrac{1}{2}\mathbf{x}^T G\mathbf{x} \Longrightarrow G = \nabla^2 f_c(\mathbf{x}) = 2C.$$

4. The investor has two competing objectives - the wish to maximize their returns while at the same time minimizing their risks. This is an example of a *multi-objective* or *vector valued* optimization problem. It may be converted into a scalar problem by taking a weighted sum of the two objectives, where the weights are non-negative and add up to 1. This gives

$$\alpha_1 f_r(\mathbf{x}) + \alpha_2 f_c(\mathbf{x}) \qquad \text{where} \quad \alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_1 + \alpha_2 = 1.$$

Using the restriction that the weights sum up to 1 to eliminate one of the weights gives the objective

$$f_\beta(\mathbf{x}) = \beta f_r(\mathbf{x}) + (1-\beta) f_c(\mathbf{x}) = -\beta \mathbf{r}^T\mathbf{x} + (1-\beta)\mathbf{x}^T C\mathbf{x}. \tag{1.1.4}$$

A weight $\beta = 0$ corresponds to minimizing the risk of the portfolio, while a weight $\beta = 1$ corresponds to minimizing the negative of the portfolio return (i.e. maximizing the portfolio return).

5. The MATLAB program in the file `popt.m` solves these problems by defining

$$\ell = \begin{bmatrix} 0.2 \\ 0.1 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} 0.8 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \end{bmatrix},$$

so

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \ell \leq \mathbf{x} \leq \mathbf{u}, A\mathbf{x} \leq \mathbf{b}\}.$$

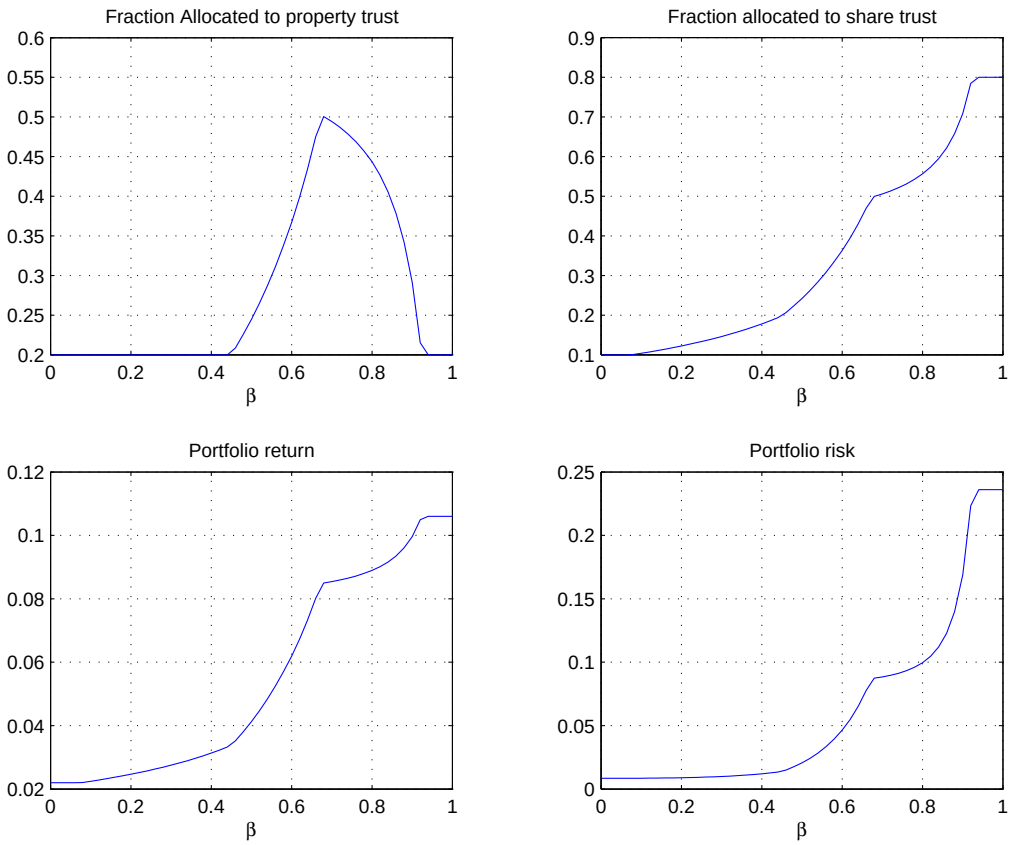


Figure 1.1.4: Asset allocation, return and risk as functions of β

Using the MATLAB linear programming routine `linprog` to maximize the returns produces the solution

$$\mathbf{x}^* = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} \qquad \text{with} \quad \mathbf{r}^T\mathbf{x} = 0.106.$$

This means that 20% of the assets should be invested in the property trust and 80% of the assets in the share trust for a return of 10.6% per annum. This solution is at a vertex of the feasible region (see Figure 1.1.3).

The objective $f_\beta(\mathbf{x})$ defined in (1.1.4) is quadratic, so can be minimized over Ω using the MATLAB quadratic programming routine `quadprog`. This was done for $\beta = 0, 0.02, 0.04, \ldots, 1$ and the solutions are plotted in Figure 1.1.3 and 1.1.4. A value $\beta = 0$ corresponds to just minimizing the risk, producing an asset allocation determined by the lower bounds of 20% in the property trust and 10% in the share trust. Not all the assets have been invested as $x_1 + x_2 = 0.3 < 1$. The corresponding return is 2.2% and risk is 0.85%. This compares with a maximal return of 10.6% with associated risk 23.6%. Values of $\beta \in (0, 1)$ gives solutions, returns and portfolio risk as illustrated in Figure 1.1.4. For β less than approximately 0.68 not all the assets are allocated. Finally Figure 1.1.5 illustrates the feasible region and objective functions $f_\beta(\mathbf{x})$ for $\beta = 0, 0.6, 1$.

Optimization methods are primarily concerned with questions such as

- Does a solution to the mathematical optimization problem exist?
- How do you recognize a solution?

  You need necessary and/or sufficient conditions (see Section 1.2.8) for a point to be a solution.
- How do you actually find a solution?

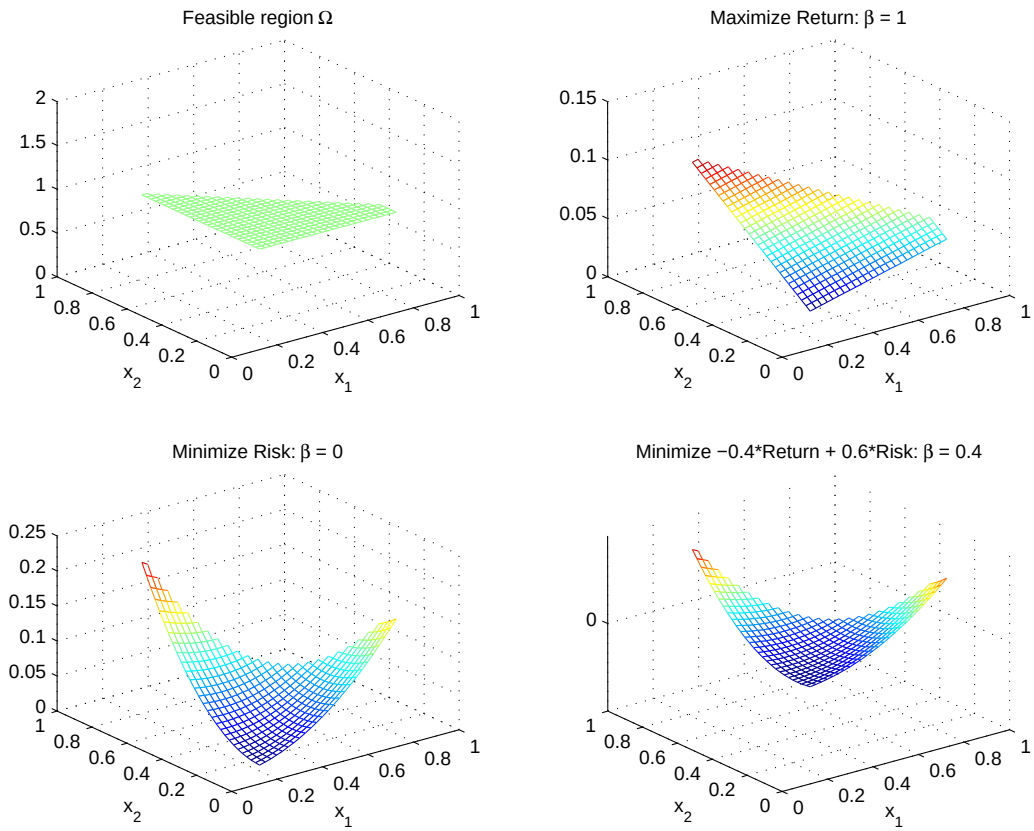


Figure 1.1.5: Objective function $f_\beta(\mathbf{x})$ over Ω

This can involve analytical methods, but more commonly it involves numerical methods. Problems can be so large that supercomputers are required to find a solution in a reasonable time.

These notes concentrate on nonlinear deterministic problems. Linear models, in particular linear programming problems, are covered in detail in many books (see [Chv83, Mur81, Sai95, Sch98, GMW91] for example). Stochastic optimization problems, where the objective function involves an expectation or the constraints are only satisfied with a certain probability, are harder to solve but can produce more realistic models (see [KW94, BL97] for example).

1.1.4 Exercises

1. An oil refinery has three different processes to produce petrol. Each process produces varying amounts of three grades of petrol: standard, super and lead-free. The production, in thousands of litres per hour, along with the cost in dollars of an hours operation for each process, are given in Table 1. The refinery must meet daily demands for 10,000 litres of standard, 30,000 litres of super

Process	Standard	Super	Lead-free	Cost
1	3	1	2	320
2	4	3	2	800
3	2	4	3	700

Table 1.1.2: Production (1,000 litres/hour) and costs ($/hour)

and 90,000 litres of lead-free petrol. How should the refinery be operated to satisfy these demands and minimize the cost?

(a) Formulate this problem as an optimization problem in standard form.

(b) What can you say about the structure of this problem?

(c) Do global extrema exist for this problem?

(d) Use the MATLAB routine `linprog` from the `optimization` toolbox to solve this problem. Interpret the solution for the oil refinery manager who is NOT a mathematician.

2. An oil company has oil wells drilled at the points A $= (1, 1)$, B $= (-1, 1)$ and C $= (0, -3)$ on a grid layout of a particular location (see Figure 1.1.6). The scale of the grid is such that one grid unit corresponds to one kilometre. The company must build pipelines from these wells to a distribution depot. The cost of the pipelines from wells A and B is \$1000/metre, but only \$500/metre from well C because of a lower flow rate from that well. For political reasons the depot must lie on the same

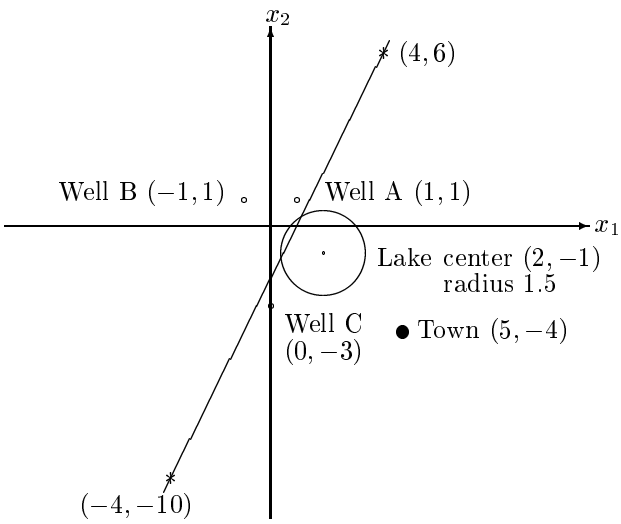


Figure 1.1.6: Grid Map for Oil Company

side of a state border as the town located at the grid point $(5, -4)$. The state border is a straight line which passes through the grid points $(-4, -10)$ and $(4, 6)$. There is also a circular lake 3.0 kilometres in diameter centred at the grid point $(2, -1)$. Obviously the depot cannot be located in the lake.

Where should the depot be constructed so as to minimize the cost of building the connecting pipelines?

(a) Pose this as a mathematical optimization problem in standard form.

(b) Say as much as you can about the structure of the problem.

(c) Solve this problem using the MATLAB routine `fmincon` from the optimization toolbox. Try starting from $\mathbf{x}^{(1)} = [\,0 \quad 1\,]^T$ and from $\mathbf{x}^{(1)} = [\,0 \quad -1\,]^T$ and interpret the results.

(d) What additions to your mathematical model may be needed to make it more realistic?

3. Given a set of data (α_i, y_i) for $i = 1, \ldots, m$ from an experiment, find the quartic polynomial

$$p(\alpha) = x_1 + x_2\alpha + x_3\alpha^2 + x_4\alpha^3 + x_5\alpha^4$$

with $p(0) \geq 0$, $p(1) \leq 1$ and $p(\alpha) \to \infty$ as $\alpha \to \infty$, which best fits the data in the sense that it minimizes the sum of squares

$$f(\mathbf{x}) = \sum_{i=1}^{m} [p(\alpha_i) - y_i]^2 = ||A\mathbf{x} - \mathbf{y}||_2^2$$

where $A_{ij} = \alpha_i^{j-1}$ for $i = 1, ..., m$ and $j = 1, ..., 5$.

(a) Show that the gradient of $f(\mathbf{x})$ is $\mathbf{g}(\mathbf{x}) = 2A^T\mathbf{r}(\mathbf{x})$ where $\mathbf{r}(\mathbf{x}) = A\mathbf{x} - \mathbf{y}$, and the Hessian of $f(\mathbf{x})$ is $\nabla^2 f(\mathbf{x}) = 2A^T A$.

(b) Show that this problem is a quadratic programming problem (i.e. the objective is a quadratic function of the variables, and all the constraints are affine functions of the variables).

(c) For the data in Table 1.1.3 use MATLAB to solve the unconstrained linear least squares problem (see `help backslash` in MATLAB). Check if any constraints are violated at the solution.

i	1	2	3	4	5	6	7	8
α_i	-3	-2	-1	0	1	2	3	4
y_i	65	8	0	1	0	10	62	230

Table 1.1.3: Data for quartic approximation

4. An investment firm has one million dollars to invest, so as to maximize its profit. The available options are:

- A construction project requiring and investment of \$480,000 and providing a profit of \$29,000 after one year. A fractional investment, as long as it is not less than 20% of the total, is possible.
- Buying units of a portfolio of stocks requiring an investment of \$19,000 per unit and yielding a profit of \$1,100 per unit per year. Only a whole number of units may be purchased.
- Buying shares of a certain stock costing \$17.50 per share and yielding a dividend of \$0.95 per share per annum.

(a) Formulate this problem as a mathematical optimization problem in standard form, and say as much as you can about its structure.

(b) Solve it by relaxing the integrality constraints and using a MATLAB routine to solve the relaxed problem. It may be necessary to add additional constraints or to modify the exisiting constraints to force the variables to have integer values where required.

1.2 Mathematical Background

This section summarizes some of the mathematical background that is needed in the study of optimization problems. Additional material on matrix algebra can be found in Appendix A.

1.2.1 Definitions of local and global extrema

Precise definitions of what is meant by a solution to an optimization problem are required. Consider the problem of optimizing the function $f : \mathbb{R}^n \to \mathbb{R}$ over the set $\Omega \subseteq \mathbb{R}^n$.

Definition 1.2.1 (Local and global minimizers)

1. *A point* $\mathbf{x}^* \in \Omega$ *is a* global minimizer *of* f *over* $\Omega \iff f(\mathbf{x}^*) \leq f(\mathbf{x})$ *for all* $\mathbf{x} \in \Omega$.

2. *A point* $\mathbf{x}^* \in \Omega$ *is a* strict global minimizer *of* f *over* $\Omega \iff f(\mathbf{x}^*) < f(\mathbf{x})$ *for all* $\mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{x}^*$.

3. *A point* $\mathbf{x}^* \in \Omega$ *is a* local minimizer *of* f *over* $\Omega \iff$ *there exists a* $\delta > 0$ *such that* $f(\mathbf{x}^*) \leq f(\mathbf{x})$ *for all* $\mathbf{x} \in \Omega$ *with* $||\mathbf{x} - \mathbf{x}^*|| \leq \delta$.

4. *A point* $\mathbf{x}^* \in \Omega$ *is a* strict local minimizer *of* f *over* $\Omega \iff$ *there exists a* $\delta > 0$ *such that* $f(\mathbf{x}^*) < f(\mathbf{x})$ *for all* $\mathbf{x} \in \Omega$ *with* $0 < ||\mathbf{x} - \mathbf{x}^*|| \leq \delta$.

Here $||\mathbf{y}||$ denote the standard Euclidean length (2-norm) of a vector in $\mathbb{R}^n$ (see Section 1.2.7 for more detail). The set $N_\delta(\mathbf{x}^*) = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x} - \mathbf{x}^*|| \leq \delta\}$ is the interior and boundary of a "ball" in $\mathbb{R}^n$ with centre $\mathbf{x}^*$ and radius δ. $N_\delta(\mathbf{x}^*)$ is also referred to as a *neighbourhood* of $\mathbf{x}^*$.

A local minimizer $\mathbf{x}^*$ has $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in some feasible neighbourhood of $\mathbf{x}^*$. In contrast a global minimizer $\mathbf{x}^*$ has $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all feasible points $\mathbf{x} \in \Omega$. Note that the point $\mathbf{x}^*$ is referred to as the *minimizer*, while the function value $f^* \equiv f(\mathbf{x}^*)$ is the *minimum*. Some books use the term absolute instead of global and relative instead of local. If a problem has a global minimum then it is the lowest of all the local minima.

The corresponding definitions for maximizers are simply obtained by reversing all the inequalities on f in Definition (1.2.1). The term *extrema* of f over Ω is used to refer collectively to all the minima and maxima of f over Ω.

It is important that the difference between local and global minimizers is clearly understood. A local minimum or maximum $f(\mathbf{x}^*)$ reflects the nature of the objective function $f(\mathbf{x})$ in a feasible neighbourhood of $\mathbf{x}^*$. In contrast a global minimum or maximum $f(\mathbf{x}^*)$ reflects the nature of the objective function over *all* of the feasible region. Unless the problem has some special properties (see for example Chapter 2 on convexity) then you have to settle for finding a local minimum. *Global optimization* refers to the problem of finding a global minimum of a function, and in general is very much more difficult than finding a local minimum. There are algorithms for trying to find a global minimum, but they involve a lot of extra work and may give only a probability that a global minimum has been found (see [HPT95, HP95, HT90] for example). Other approaches to global optimization are based on simulated annealing [KGV83], genetic algorithms [Mic94, Gol89], and interval analysis [Han92].

Example 1.2.2 (Local and global extrema) *Let*

$$f(x) = \begin{cases} (x-1)^2 & \text{if } x \leq 0.5; \\ 0.25 & \text{if } 0.5 < x \leq 1; \\ 1.25 - (x-2)^2 & \text{if } 1 < x \leq 2.5; \\ x - 1.5 & \text{if } 2.5 < x \leq 3; \\ 1.5 - 0.25\sin(\pi(x-3)) & \text{if } 3 < x. \end{cases} \tag{1.2.1}$$

The function (1.2.1) is plotted in Figure 1.2.1. The points $0, 0.5, 2, 2.5, 3, 3.5, 4.5, 5$ *are extrema of the function* $f(x)$ *on the interval* $\Omega = [0, 5]$, *In particular*

a) $x^{(a)} = 0$ *is strict local maximizer with* $f^{(a)} = 1$;

b) *any point* $x^{(b)}$ *in the interval* $[0.5, 1]$ *is a local and global minimizer with* $f^{(b)} = 0.25$ *(but not strict as adjacent points have the same function value);*

c) $x^{(c)} = 2$ *is a strict local maximizer with* $f^{(c)} = 1.25$;

d) $x^{(d)} = 2.5$ *is a strict local minimizer with* $f^{(d)} = 1$;

e) $x^{(e)} = 3$ *is a strict local maximizer with* $f^{(e)} = 1.5$;

f) $x^{(f)} = 3.5$ *is a strict local minimizer with* $f^{(f)} = 1.25$;

g) $x^{(g)} = 4.5$ *is a strict local and global maximizer with* $f^{(g)} = 1.75$;

h) $x^{(h)} = 5$ *is a strict local minimizer with* $f^{(h)} = 1.5$.

A special case when a global minimum or global maximum can be found is when the function has a known bound and a point achieving that bound is found. For instance

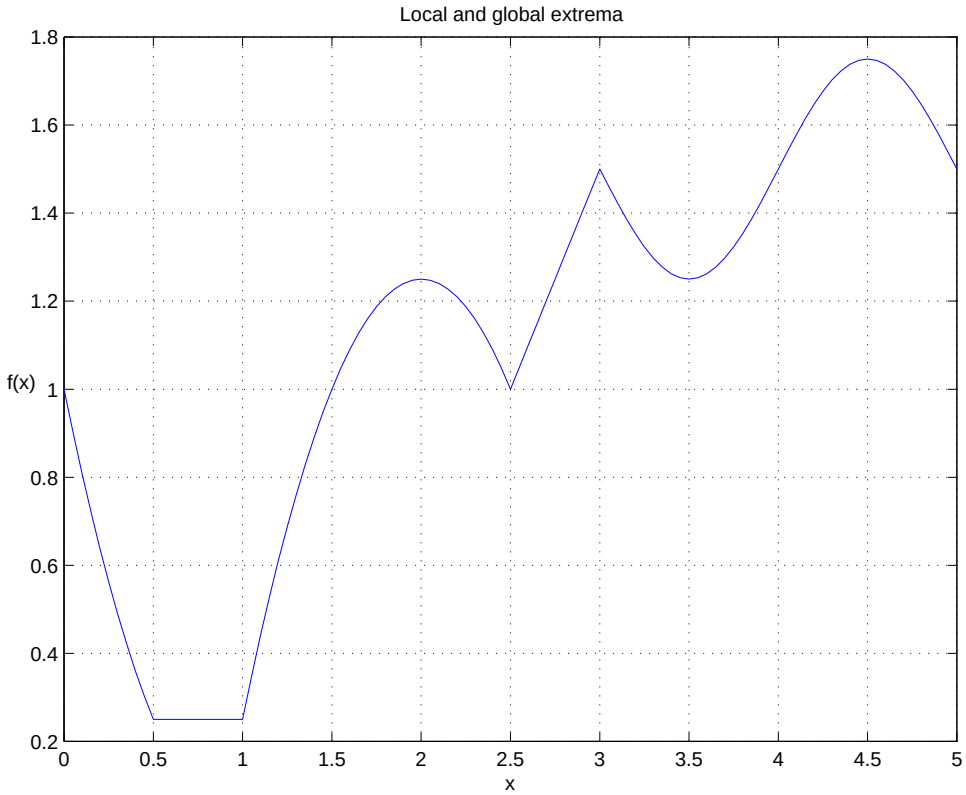


Figure 1.2.1: Local and global extrema

- $f(\mathbf{x}) \geq \eta \ \forall \mathbf{x} \in \Omega$ and $\overline{\mathbf{x}} \in \Omega$ has $f(\overline{\mathbf{x}}) = \eta \Longrightarrow \overline{\mathbf{x}}$ is a global minimizer of f over Ω.
- $f(\mathbf{x}) \leq \zeta \ \forall \mathbf{x} \in \Omega$ and $\hat{\mathbf{x}} \in \Omega$ has $f(\hat{\mathbf{x}}) = \zeta \Longrightarrow \hat{\mathbf{x}}$ is a global minimizer of f over Ω.

A common example is when a function $f : \mathbb{R}^n \to \mathbb{R}$ can be expressed as a sum of squares

$$f(\mathbf{x}) = \sum_{i=1}^{m} [r_i(\mathbf{x})]^2,$$

so $f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Then any point $\mathbf{x}^*$ such that $f(\mathbf{x}^*) = 0$ is a global minimizer of $f(\mathbf{x})$. Methods for minimizing sums of squares are discussed in Section 5.5 of Chapter 5.

1.2.2 Existence of extrema

Proposition 1.2.3 *If Ω is a compact set in $\mathbb{R}^n$ and f is continuous on Ω then the global extrema of f on Ω exist.*

In $\mathbb{R}^n$ a set Ω is compact if and only if it is closed and bounded. A set $\Omega \subseteq \mathbb{R}^n$ is

- *closed* if for any sequence $\{\mathbf{x}^{(k)}\}$ such that $\mathbf{x}^{(k)} \in \Omega$ for all k and $\mathbf{x}^{(k)} \to \mathbf{x}^*$ as $k \to \infty$ then $\mathbf{x}^* \in \Omega$.
- *bounded* if there exists a finite constant K such that

$$\|\mathbf{x}\| \leq K \text{ for all } \mathbf{x} \in \Omega.$$

For example $\Omega = [0, \infty)$ is closed, but not bounded, while $\Omega = [0, 10)$ is bounded, but not closed.

If any of the conditions in Proposition 1.2.3 are not satisfied then global extrema may or may not exist. Proposition 1.2.3 only provides sufficient conditions which guarantee the existence of both a global minimizer and a global maximizer.

The continuity of f can be relaxed by considering *upper semi-continuous* or *lower semi-continuous* functions. More information can be found in [Wou79, page 80] or another book on Functional Analysis. In particular care must be taken in infinite dimensional spaces, for example the space of continuous functions on $[0, 1]$.

Example 1.2.4 (Existence of global extrema)
The following examples illustrate the application of Proposition 1.2.3

1. *Let $\Omega = [1, \infty)$ and $f(x) = \sin(x)$. The set Ω is not bounded, but f has a global minimum of -1 with global minimizers $-\frac{\pi}{2} + 2k\pi$ for any integer $k \geq 1$, and a global maximum of $+1$ with global maximizers $\frac{\pi}{2} + 2k\pi$ for any integer $k \geq 0$.*

2. *Let $\Omega = (-2, -3) = \{x \in \mathbb{R} : -2 < x < 3\}$ and $f(x) = x$. The set Ω is bounded, but not closed. This problem has neither a global minimum nor a global maximum.*

3. *Let $\Omega = [0, 1] = \{x \in \mathbb{R} : 0 \leq x \leq 1\}$ and $f(x) = \log(x)$. The set Ω is compact as it is closed and bounded. This problem has a global maximum of 0 with global maximizer $x^* = 1$. However it does not have a global minimum as $f(x) \to -\infty$ as $x \to 0^+$.*

4. *Let $\Omega = [1, 2] = \{x \in \mathbb{R} : 1 \leq x \leq 2\}$ and $f(x) = \log(x)$. The set Ω is compact as it is closed and bounded, and f is continuous on Ω, so global extrema exist. This problem has a global maximum of $\log(2)$ attained at $x = 2$, and a global minimum of 0 attained at $x = 1$.*

Even if the global minimum does not exist it may still be possible to find the "best" lower bound on the function $f(\mathbf{x})$ over the set Ω. Similarly it may be possible to find the best upper bound.

Definition 1.2.5 (H) *The* infimum *of $f : \mathbb{R}^n \to \mathbb{R}$ over $\Omega \subseteq \mathbb{R}^n$ is the largest real number ζ such that*

$$f(\mathbf{x}) \geq \zeta \quad \textit{for all } \mathbf{x} \in \Omega.$$

Similarly the supremum *of f over Ω is the smallest real number ξ such that*

$$f(\mathbf{x}) \leq \xi \quad \textit{for all } \mathbf{x} \in \Omega.$$

These are often written

$$\zeta = \inf_{\mathbf{x} \in \Omega} f(\mathbf{x}) \quad \text{and} \quad \xi = \sup_{\mathbf{x} \in \Omega} f(\mathbf{x}).$$

The infimum and supremum are not necessarily achieved at any point in Ω, for example $\inf_{\mathbf{x} \in \mathbb{R}} e^{-x} = 0$. If the global minimum exists then the infimum is equal to the global minimum. Similarly if the global maximum exists then the supremum is equal to the global maximum.

1.2.3 Gradients and Hessians

Classically optimization problems are formulated in terms of smooth functions, that is functions which are at least once continuously differentiable. The notation $f \in C^k(\mathbb{R}^n)$ denotes a function f whose first k partial derivatives are continuous functions defined on $\mathbb{R}^n$. When $f \in C^1$ the *gradient* is the continuous vector valued function $\nabla f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ consisting of the first partial derivatives of f with respect to the variables $\mathbf{x}$. Frequently the notation $\mathbf{g}(\mathbf{x})$ is used for the gradient, so

$$\mathbf{g}(\mathbf{x}) \equiv \nabla f(\mathbf{x}) \equiv \begin{bmatrix} \dfrac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}.$$

When $f \in C^2$ the *Hessian* is the continuous matrix valued function $\nabla^2 f(\mathbf{x})$ consisting of second partial derivatives of f with respect to the variables $\mathbf{x}$. Frequently the notation $G(\mathbf{x})$ is used for $\nabla^2 f(\mathbf{x})$, so

$$G(\mathbf{x}) \equiv \nabla^2 f(\mathbf{x}) \equiv \begin{bmatrix} \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix}.$$

As there are n variables $\mathbf{x}$ the gradient $\mathbf{g}(\mathbf{x})$ is a column vector with n components, and the Hessian $G(\mathbf{x})$ is an n by n matrix. Both the gradient $\mathbf{g}(\mathbf{x})$ and the Hessian $G(\mathbf{x})$ depend upon the variables $\mathbf{x}$. If $f \in C^2$ then the Hessian matrix is symmetric, that is $G^T(\mathbf{x}) = G(\mathbf{x})$ so

$$G_{ij}(\mathbf{x}) \equiv \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i} \equiv G_{ji}(\mathbf{x}).$$

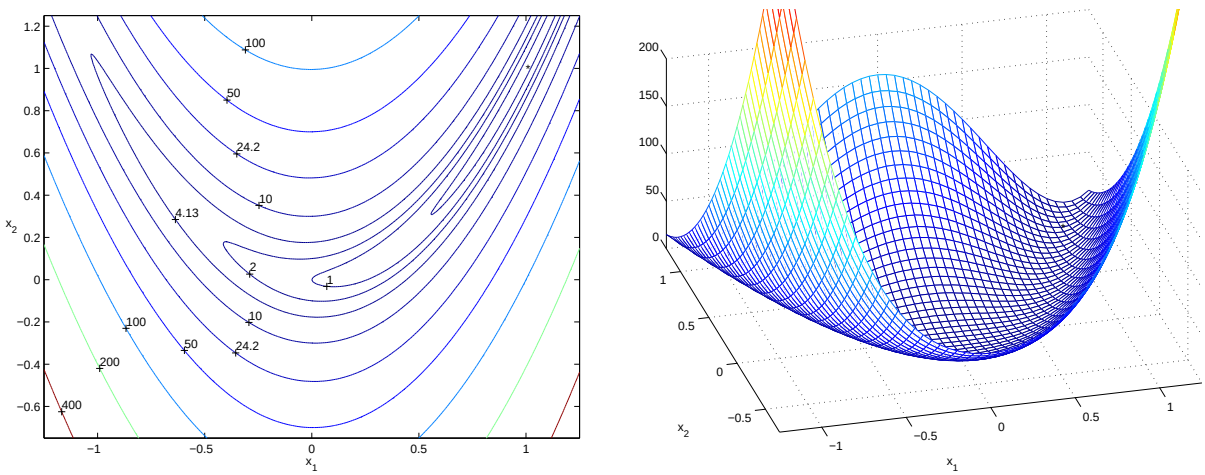


Figure 1.2.2: Contour and mesh plots of Rosenbrock's function

Example 1.2.6 (Rosenbrock's function)

Rosenbrock's function is a well known test problem in unconstrained optimization defined by

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

As this function is simply a polynomial in the variables x_1 and x_2 it is infinitely differentiable. Its gradient is

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

and its Hessian is

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}.$$

The behaviour of functions of two variables can be illustrated by contour plots or surface plots. A *contour* of a function $f : \mathbb{R}^n \to \mathbb{R}$ is a set $\{\, \mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) = \eta \,\}$, where $\eta \in \mathbb{R}$. Figure 1.2.2 gives the contours and a 3-D view of Rosenbrock's function. It is a steep sided valley, and the bottom of the valley slowly decreases to a minimum at the point $\mathbf{x}^* = [1\ 1]^T$.

The gradient $\nabla f(\overline{\mathbf{x}})$ at a point $\overline{\mathbf{x}}$ is orthogonal to the contour $\{\, \mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) = f(\overline{\mathbf{x}}) \,\}$ passing through $\overline{\mathbf{x}}$. A vector $\mathbf{v} \in \mathbb{R}^n$ is is tangential to the contour at $\overline{\mathbf{x}}$ if and only if $\mathbf{v}^T \nabla f(\overline{\mathbf{x}}) = 0$. Let $\overline{\mathbf{x}}$ be a point

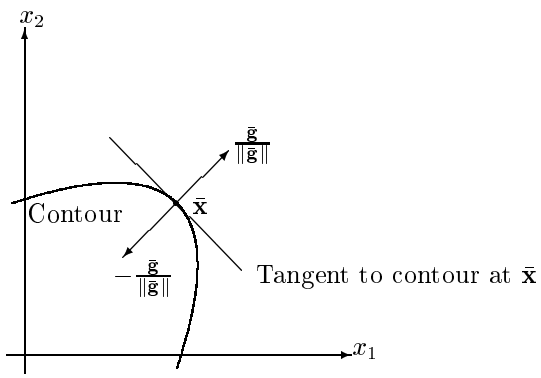


Figure 1.2.3: Directions of greatest and least slope

where the gradient $\bar{\mathbf{g}} = \nabla f(\overline{\mathbf{x}}) \neq 0$. The direction $\mathbf{d} = \bar{\mathbf{g}}/\|\bar{\mathbf{g}}\|$ is the direction of greatest slope at $\overline{\mathbf{x}}$, while $\mathbf{d} = -\bar{\mathbf{g}}/\|\bar{\mathbf{g}}\|$ is the direction of least slope at $\overline{\mathbf{x}}$, over all directions $\mathbf{d}$ with $\|\mathbf{d}\| = 1$. This is illustrated in Figure 1.2.3.

Any **affine** function (often also referred to a a *linear* function) can be written as

$$f(\mathbf{x}) = \mathbf{g}^T\mathbf{x} + f_0,$$

where $\mathbf{g}$ is a fixed vector in $\mathbb{R}^n$ and f_0 is a scalar. Thus

$$f_0 = f(\mathbf{0}), \quad \nabla f(\mathbf{x}) = \mathbf{g} \quad \text{and} \quad \nabla^2 f(\mathbf{x}) = \mathbf{0}.$$

Any **quadratic** function $f : \mathbb{R}^n \to \mathbb{R}$ can be written as

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{g}_0^T\mathbf{x} + f_0,$$

where G is a fixed $n \times n$ real matrix, $\mathbf{g}_0$ is a fixed n-dimensional vector and f_0 is a fixed scalar. Then

$$\nabla f(\mathbf{x}) = \tfrac{1}{2}(G + G^T)\mathbf{x} + \mathbf{g}_0 \quad \text{and} \quad \nabla^2 f(\mathbf{x}) = \tfrac{1}{2}(G + G^T),$$

so $\nabla f(0) = \mathbf{g}_0$. A quadratic function can always be written with a symmetric matrix G. If G is not symmetric simply replace G by $\frac{1}{2}(G + G^T)$. If G is symmetric then

$$\nabla f(\mathbf{x}) = G\mathbf{x} + \mathbf{g}_0 \quad \text{and} \quad \nabla^2 f(\mathbf{x}) = G.$$

1.2.4 Vectors and lines in $\mathbb{R}^n$

The set of points

$$\mathbf{x}(\alpha) = \bar{\mathbf{x}} + \alpha\mathbf{d} \quad \alpha \in \mathbb{R} \tag{1.2.2}$$

describes a line in $\mathbb{R}^n$ passing through the point $\bar{\mathbf{x}}$ and lying in the direction $\mathbf{d}$. If $\alpha \geq 0$ then (1.2.2) represents a ray starting at $\bar{\mathbf{x}}$ going in the direction $\mathbf{d}$. If α is restricted to lie in an interval, say $\alpha \in [a, b]$ where $a < b$ are real numbers, then (1.2.2) represents the line segment joining $\bar{\mathbf{x}} + a\mathbf{d}$ and $\bar{\mathbf{x}} + b\mathbf{d}$. Alternatively the line segment joining $\mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ has the parametric representation

$$\mathbf{x}(\theta) = \theta\mathbf{y} + (1 - \theta)\mathbf{z} \quad \text{for } \theta \in [0, 1]. \tag{1.2.3}$$

Often it is convenient to normalize the direction vector $\mathbf{d}$ so that $\|\mathbf{d}\|_2 = 1$, or equivalently $\mathbf{d}^T\mathbf{d} = 1$. Then $\mathbf{d}$ just signifies the direction, the line is exactly the same, but the scalar corresponding to a given point will change.

The function value along the line (1.2.2) is $\ell(\alpha) = f(\mathbf{x}(\alpha))$, which is a function of a single variable. The slope along the line at the point $\mathbf{x}(\alpha)$ is then

$$\ell'(\alpha) = \frac{df(\mathbf{x}(\alpha))}{d\alpha} = \nabla f(\mathbf{x}(\alpha))^T\mathbf{d}. \tag{1.2.4}$$

and the curvature along the line is

$$\ell''(\alpha) = \frac{d^2 f(\mathbf{x}(\alpha))}{d\alpha^2} = \mathbf{d}^T \nabla^2 f(\mathbf{x}(\alpha))^T \mathbf{d}. \tag{1.2.5}$$

Example 1.2.7 (Line in $\mathbb{R}^2$, slope and curvature) *Let*

$$\bar{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 5 \\ 3 \end{bmatrix},$$

The normalized direction vector with length 1 *is*

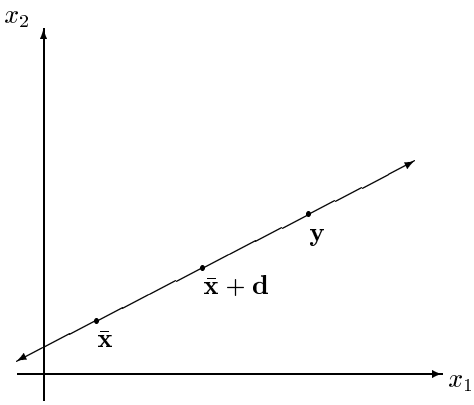


Figure 1.2.4: Lines in $\mathbb{R}^2$

$$\bar{\mathbf{d}} = \frac{\mathbf{d}}{\|\mathbf{d}\|} = \frac{1}{\sqrt{5}}\mathbf{d} = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}.$$

Expressing $\mathbf{y} = \bar{\mathbf{x}} + \bar{\alpha}\bar{\mathbf{d}}$ *gives*

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \bar{\alpha}\frac{1}{\sqrt{5}}\begin{bmatrix} 2 \\ 1 \end{bmatrix} \iff \bar{\alpha} = 2\sqrt{5}.$$

Rosenbrock's function (Example 1.2.6) has

$$f(\bar{\mathbf{x}}) = 0, \quad \nabla f(\bar{\mathbf{x}}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textit{and} \quad \nabla^2 f(\bar{\mathbf{x}}) = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}.$$

Along the line

$$\mathbf{x}(\alpha) = \bar{\mathbf{x}} + \alpha\bar{\mathbf{d}} = \begin{bmatrix} 1 + \frac{2\alpha}{\sqrt{5}} \\ 1 + \frac{\alpha}{\sqrt{5}} \end{bmatrix}$$

the function value is

$$\ell(\alpha) \equiv f(\mathbf{x}(\alpha)) = 64\alpha^4 + 96\sqrt{5}\alpha^3 + \frac{904}{5}\alpha^2.$$

The first derivative along the line is

$$\begin{aligned} \bar{\mathbf{d}}^T \nabla f(\bar{\mathbf{x}} + \alpha\bar{\mathbf{d}}) &= \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix}^T \begin{bmatrix} 128\sqrt{5}\alpha^3 + 800\alpha^2 + \frac{1204}{\sqrt{5}}\alpha \\ -160\alpha^2 - 120\sqrt{5}\alpha \end{bmatrix} \\ &= 256\alpha^3 + 288\sqrt{5}\alpha^2 + \frac{1808}{5}\alpha \\ &= \ell'(\alpha), \end{aligned}$$

which is the slope of f at the point $\bar{\mathbf{x}} + \alpha\bar{\mathbf{d}}$ in the direction $\bar{\mathbf{d}}$. The second derivative along the line is

$$\begin{aligned}
\bar{\mathbf{d}}^T\nabla^2 f(\bar{\mathbf{x}} + \alpha\bar{\mathbf{d}})\bar{\mathbf{d}} &= \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix}^T \begin{bmatrix} 960\alpha^2 + 880\sqrt{5}\alpha + 802 & -160\sqrt{5}\alpha \\ -160\sqrt{5}\alpha - 400 & 200 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} \\
&= 768\alpha^2 + 576\sqrt{5}\alpha + \frac{1808}{5} \\
&= \ell''(\alpha),
\end{aligned}$$

which is the curvature of f at the point $\bar{\mathbf{x}} + \alpha\bar{\mathbf{d}}$ in the direction $\bar{\mathbf{d}}$.

1.2.5 Taylor series

Local properties of a multi-variable function are often characterized by multi-variable Taylor series. If $f \in C^1(\mathbb{R}^n)$ then first order Taylor series expansion of f about a point $\mathbf{x}^*$ is

$$f(\mathbf{x}^* + \alpha\mathbf{d}) = f(\mathbf{x}^*) + \alpha\mathbf{d}^T\nabla f(\mathbf{x}^*) + o(\alpha\|\mathbf{d}\|). \tag{1.2.6}$$

Remember that

$$\beta(\alpha) = o(\gamma(\alpha)) \iff \lim_{\alpha\to 0^+} \frac{\beta(\alpha)}{\gamma(\alpha)} = 0 \tag{1.2.7}$$

and

$$\beta(\alpha) = O(\gamma(\alpha)) \iff \lim_{\alpha\to 0^+} \frac{\beta(\alpha)}{\gamma(\alpha)} = K. \tag{1.2.8}$$

where K is a non-zero constant. If $f \in C^2(\mathbb{R}^n)$ then the $o(\alpha\|\mathbf{d}\|)$ term in (1.2.6) can be replaced by a $O(\alpha^2\|\mathbf{d}\|^2)$ term.

Example 1.2.8 (Order notation) *Examples of the use of order notation as $\alpha \to 0$ are*

$$\alpha^2 = o(\alpha), \quad \alpha^3 = o(\alpha^2), \quad \alpha^{1+\epsilon} = o(\alpha) \text{ and } \alpha^{2+\epsilon} = o(\alpha^2)$$

for any $\epsilon > 0$. Also

$$\beta = 76\alpha^4 - 96\alpha^3 + 1001\alpha^2 \Longrightarrow \beta = O(\alpha^2), \text{ and } \beta = o(\alpha).$$

If $f \in C^2(\mathbb{R}^n)$ then the second order Taylor series expansion of f about a point $\mathbf{x}^*$ is

$$f(\mathbf{x}^* + \alpha\mathbf{d}) = f(\mathbf{x}^*) + \alpha\mathbf{d}^T\nabla f(\mathbf{x}^*) + \tfrac{1}{2}\alpha^2\mathbf{d}^T\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha^2\|\mathbf{d}\|^2). \tag{1.2.9}$$

If $f \in C^3(\mathbb{R}^n)$ then the $o(\alpha^2\|\mathbf{d}\|^2)$ term in (1.2.9) can be replaced by a $O(\alpha^3\|\mathbf{d}\|^3)$ term. The first order expansion of the gradient ∇f about a point $\mathbf{x}^*$ is

$$\nabla f(\mathbf{x}^* + \alpha\mathbf{d}) = \nabla f(\mathbf{x}^*) + \alpha\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha\|\mathbf{d}\|). \tag{1.2.10}$$

The little o and big O notation can also be used describe the rate at which a sequence $\{\mathbf{x}^{(k)}\}$ approaches its limit point (see section 1.5.2).

1.2.6 Positive definite matrices

If a function $f(x)$ of a single variable $x \in \mathbb{R}$ has a stationary point x^* with $f'(x^*) = 0$ and the second derivative $f''(x^*) > 0$ then x^* is a local minimizer of f. In multi-variable optimization the second derivatives form the n by n symmetric Hessian matrix. Chapter 3 shows that the corresponding condition on the Hessian that guarantees a stationary point is a local minimizer is that the Hessian is positive definite. The concepts of positive definite and negative definite matrices generalize to symmetric matrices the idea of positive and negative numbers, and play a key role in multi-variable nonlinear optimization.

Definition 1.2.9 *Let G be an n by n real symmetric matrix. Then*

1. *G is positive definite* $\iff$ $\mathbf{x}^T G\mathbf{x} > 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \neq 0$.
2. *G is positive semi-definite* $\iff$ $\mathbf{x}^T G\mathbf{x} \geq 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$.
3. *G is negative definite* $\iff$ $\mathbf{x}^T G\mathbf{x} < 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \neq 0$.
4. *G is negative semi-definite* $\iff$ $\mathbf{x}^T G\mathbf{x} \leq 0$ *for all* $\mathbf{x} \in \mathbb{R}^n$.
5. *G is indefinite* $\iff$ *there exists* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *such that* $\mathbf{x}^T G\mathbf{x} > 0$ *and* $\mathbf{y}^T G\mathbf{y} < 0$.

Note that G is negative (semi-)definite if and only if $-G$ is positive (semi-) definite. Also a matrix with all positive elements, that is $G_{ij} > 0$ for all i, j, is not necessarily positive definite.

For small matrices ($n = 2$ or $n = 3$) characterizations of positive definiteness in terms of the principal minors can be used. Remember that the ith principal minor of G is the determinant of the leading i by i submatrix of G (see (A.3.1) in Appendix A).

1. G is positive definite $\iff$ *all* principal minors are positive.
2. G is negative definite $\iff$ the ith principal minor has sign -1^i.
3. If $n \geq 2$ and all the principal minors are all negative then G is indefinite.

A real n by n symmetric matrix has n real eigenvalues $\lambda_i(G), i = 1, \ldots, n$. This leads to the following characterizations

1. G is positive definite $\iff$ $\lambda_i(G) > 0$ for $i = 1, \ldots, n$.
2. G is positive semi-definite $\iff$ $\lambda_i(G) \geq 0$ for $i = 1, \ldots, n$.
3. G is negative definite $\iff$ $\lambda_i(G) < 0$ for $i = 1, \ldots, n$.
4. G is negative semi-definite $\iff$ $\lambda_i(G) \leq 0$ for $i = 1, \ldots, n$.
5. G is indefinite $\iff$ there exist $\lambda_i(G) > 0$ and $\lambda_j(G) < 0$.

Also

$$\operatorname{trace} G \equiv \sum_{i=1}^{n} G_{ii} = \sum_{i=1}^{n} \lambda_i(G) \tag{1.2.11}$$

$$\det G = \prod_{i=1}^{n} \lambda_i(G). \tag{1.2.12}$$

For a 2 by 2 symmetric matrix

$$G = \begin{bmatrix} \alpha & \beta \\ \beta & \gamma \end{bmatrix}$$

the sum of the eigenvalues is $\operatorname{trace} G = \lambda_1(G) + \lambda_2(G) = \alpha + \gamma$. The product of the eigenvalues is $detG = \lambda_1(G)\lambda_2(G) = \alpha\gamma - \beta^2$. Thus any 2 by 2 matrix with positive determinant is either positive definite or negative definite. Then if $\det G > 0$ and $\operatorname{trace} G > 0$ the matrix G is positive definite, while if $\det G > 0$ and $\operatorname{trace} G < 0$ the matrix G is negative definite.

A diagonal matrix

$$D = \operatorname{diag}(\alpha_1, \ldots, \alpha_n) = \begin{bmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{bmatrix}$$

has eigenvalues equal to its diagonal elements, i.e. $\lambda_i(D) = \alpha_i$ for $i = 1, \ldots, n$. Thus a diagonal matrix is

1. positive definite $\iff$ all its diagonal elements are positive,
2. positive semi-definite $\iff$ all its diagonal elements are non-negative,

3. negative definite $\iff$ all its diagonal elements are negative,

4. negative semi-definite $\iff$ all its diagonal elements are non-positive,

5. indefinite $\iff$ some diagonal elements are positive and some are negative.

These characterizations of positive definiteness in terms of just the diagonal elements of a matrix *only* apply to a diagonal matrix in which all the off-diagonal elements are zero.

For large n the numerically most efficient way of determining if a symmetric matrix is positive definite is to try to calculate its *Cholesky factorization* (see Section A.7.2 in Appendix A). A symmetric matrix G is positive definite if and only if there exists a lower triangular matrix L such that $G = LL^T$.

Example 1.2.10 (Quadratic functions in $\mathbb{R}^2$) *Figures 1.2.5 and 1.2.6 give the contours and surface plots of the quadratic functions $f : \mathbb{R}^2 \to \mathbb{R}$ defined by*

$$f_1(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2, \;\; G = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \;\; \textit{eigenvalues} \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \; G \textit{ positive definite}$$

$$f_2(\mathbf{x}) = x_1^2 + x_2^2 + 3x_1x_2, \;\; G = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}, \;\; \textit{eigenvalues} \begin{bmatrix} -5 \\ 1 \end{bmatrix}, \; G \textit{ indefinite}$$

$$f_3(\mathbf{x}) = -x_1^2 - x_2^2 + x_1x_2, \;\; G = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}, \;\; \textit{eigenvalues} \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \; G \textit{ negative definite}$$

$$f_4(\mathbf{x}) = x_1^2 - x_2^2 - x_1x_2, \;\; G = \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix}, \;\; \textit{eigenvalues} \begin{bmatrix} -\sqrt{5} \\ \sqrt{5} \end{bmatrix}, \; G \textit{ indefinite.}$$

All these quadratics have

$$\mathbf{g}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textit{and} \quad f_0 = 0.$$

If $\lambda_1 \geq \lambda_2 > 0$ are the eigenvalues of the Hessian matrix of a quadratic function $f : \mathbb{R}^2 \to \mathbb{R}$ and $\mathbf{v}_1, \mathbf{v}_2$ are the corresponding eigenvectors then the contour (or level set)

$$\{ \mathbf{x} \in \mathbb{R}^2 : f(\mathbf{x}) = 1 \}$$

is an ellipse with axes in the directions of the eigenvectors and focal lengths $1/\lambda_1$ and $1/\lambda_2$. For the first quadratic $f^{(a)}(\mathbf{x})$ in Figure 1.2.5

$$\lambda_1^{(a)} = 3, \mathbf{v}_1^{(a)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \lambda_2^{(a)} = 1, \mathbf{v}_2^{(a)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

1.2.7 Vector norms

If $\alpha \in \mathbb{R}$ then its magnitude (or absolute value) $|\alpha|$ is defined by

$$|\alpha| = \begin{cases} \alpha & \text{if } \alpha \geq 0; \\ -\alpha & \text{if } \alpha \leq 0. \end{cases}$$

However in $\mathbb{R}^n$ and $\mathbb{R}^{m \times n}$ there are several possible definitions of the magnitude of a quantity. These are referred to as vector norms and matrix norms and are denoted using $|| \; ||$.

A **vector norm** on $\mathbb{R}^n$ is a function $||.||$ from $\mathbb{R}^n$ to $\mathbb{R}$ with the following properties:

1. $||\mathbf{x}|| \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$ and $||\mathbf{x}|| = 0 \iff \mathbf{x} = \mathbf{0}$.

2. $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. (Triangle inequality)

3. $||\alpha\mathbf{x}|| = |\alpha| \, ||\mathbf{x}||$ for all $\alpha \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$.

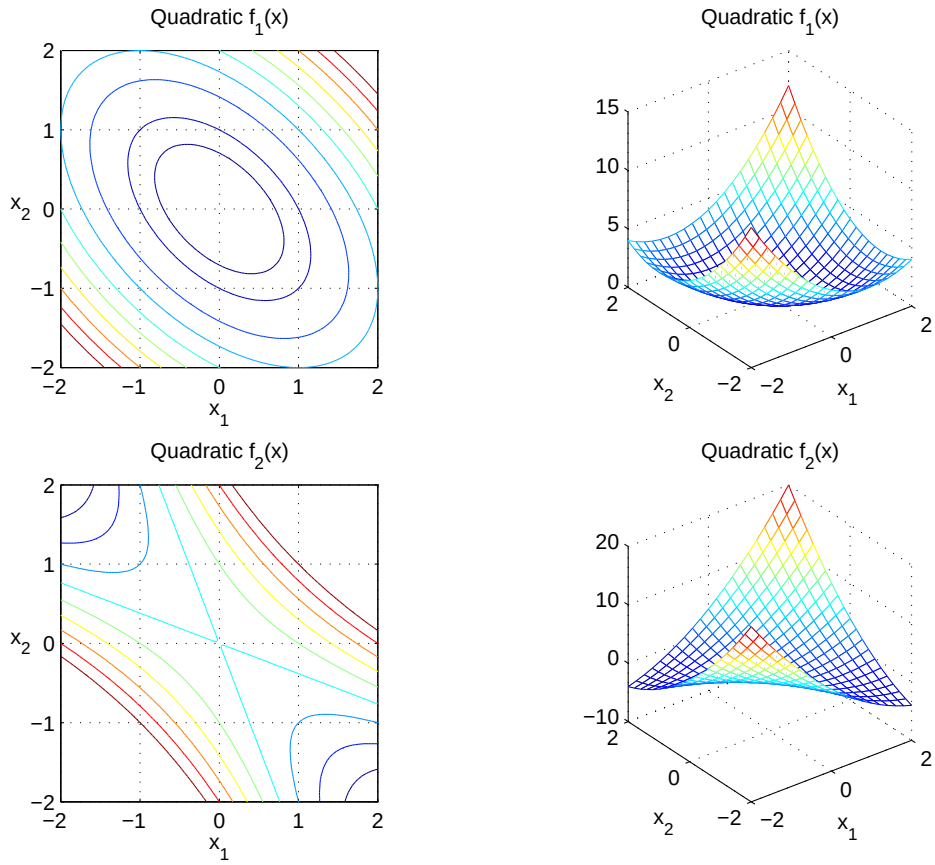


Figure 1.2.5: Quadratic functions

The most widely used vector norms are the p-norms defined by

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

and in particular the

1. **1-norm**

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|$$

2. **2-norm**

$$||\mathbf{x}||_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} = (\mathbf{x}^T\mathbf{x})^{\frac{1}{2}}$$

3. ∞**-norm** or maximum norm

$$||\mathbf{x}||_\infty = \max_{i=1,\ldots,n} |x_i|.$$

Whenever there is no subscript on the $||.||$ the 2-norm is being used unless otherwise indicated. Note that $||\mathbf{x}||_2^2 = \mathbf{x}^T\mathbf{x}$, and that the 2-norm satisfies the Cauchy-Schwarz inequality

$$|\mathbf{x}^T\mathbf{y}| \leq ||\mathbf{x}||_2\, ||\mathbf{y}||_2.$$

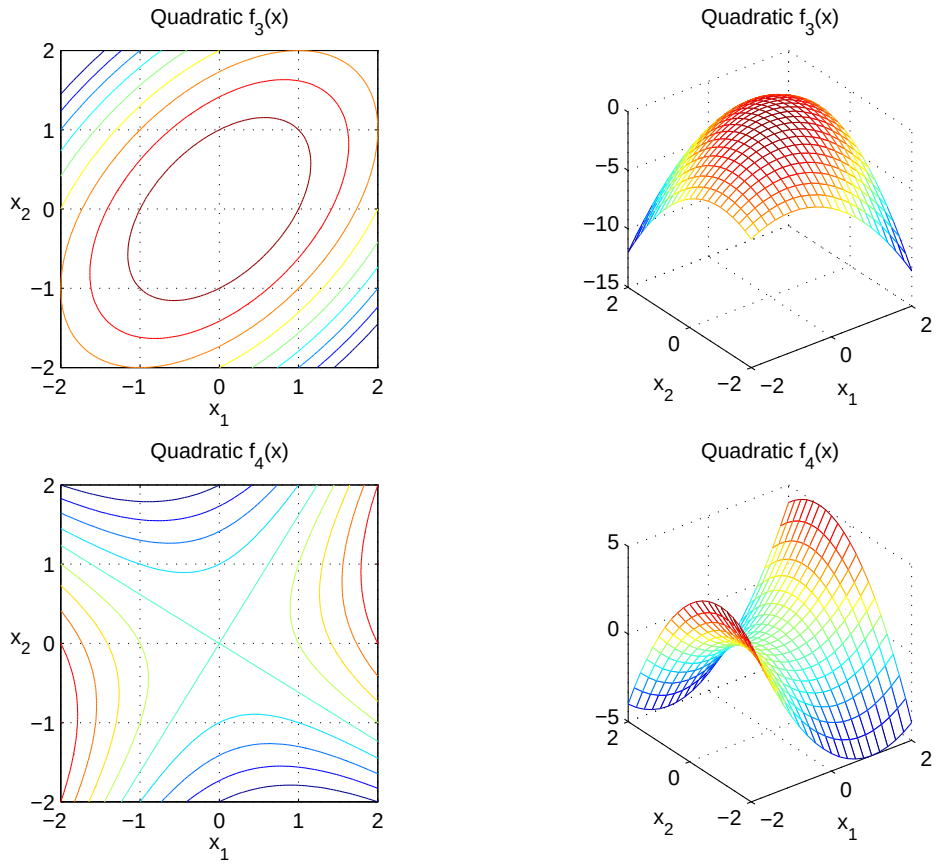


Figure 1.2.6: Quadratic functions

The 2-norm is invariant under orthogonal transformations, that is if $Q^TQ = I$ then $||Q\mathbf{x}||_2 = ||\mathbf{x}||_2$.

A useful generalization of the 2-norm is the **A-norm** defined by

$$||\mathbf{x}||_A = (\mathbf{x}^T A\mathbf{x})^{\frac{1}{2}}$$

where A is a positive definite matrix.

Some idea of the characteristics of these norms can be obtained from their unit balls

$$N = \{\, \mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}|| \leq 1 \,\}.$$

Examples of the unit balls in R^2 for the 1-norm, 2-norm, ∞-norm, and A-norm are given in Example A.6.1 on page 234 of Appendix A. Although norms are continuous functions, the 1-norm and ∞-norms are not continuously differentiable. In particular the 1-norm $||\mathbf{x}||_1$ is not continuously differentiable at any point with $x_i = 0$ for some i, while the ∞-norm $||\mathbf{x}||_\infty$ is not continuously differentiable at any point where $|x_i| = ||\mathbf{x}||_\infty$ for more than one index i. The 2-norm $||\mathbf{x}||_2$ is not continuously differentiable at $\mathbf{x} = 0$. In problems which involve minimizing the 2-norm this problem is commonly avoided by minimizing the sum of squares $||\mathbf{x}||_2^2 = \mathbf{x}^T\mathbf{x}$, which is continuously differentiable.

Norms provide the concept of distance between elements of $\mathbb{R}^n$ and are used in the definitions of continuity and convergence. The sequence $\{\mathbf{x}^{(k)}\}$ converges to $\mathbf{x}^* \in \mathbb{R}^n$ if and only if $||\mathbf{x}^{(k)} - \mathbf{x}^*|| \to 0$ as $k \to \infty$.

Suppose $\hat{\mathbf{x}} \in \mathbb{R}^n$ is an approximation to $\mathbf{x} \in \mathbb{R}^n$. For a given norm the **absolute error** in $\hat{\mathbf{x}}$ is

$$\epsilon_a = ||\hat{\mathbf{x}} - \mathbf{x}||$$

and if $x \neq 0$ the **relative error** is

$$\epsilon_r = \frac{||\hat{\mathbf{x}} - \mathbf{x}||}{||\mathbf{x}||}.$$

The relative error using the ∞-norm gives some information about the number of correct significant digits in $\mathbf{x}$. In particular if

$$\frac{||\hat{\mathbf{x}} - \mathbf{x}||_\infty}{||\mathbf{x}||_\infty} \approx 10^{-p}$$

then the largest component of $\hat{\mathbf{x}}$ has approximately p correct significant digits. For example if $\mathbf{x} = [\ -3.641 \quad 0.7843\ ]^T$ and $\hat{\mathbf{x}} = [\ -3.633 \quad 0.7915\ ]^T$, then $||\hat{\mathbf{x}} - \mathbf{x}||/||\mathbf{x}||_\infty \approx 0.002 < 0.5 \times 10^{-2}$. Here $\hat{x}_1$ has two correct significant digits while $\hat{x}_2$ has only one correct significant digit.

Example 1.2.11 (Nonlinear Data fitting) *The Caelli data in Table 1.2.1 and the approximating functions*

i	t_i	y_i	i	t_i	y_i	i	t_i	y_i	i	t_i	y_i
1	0.0	1.366	18	1.7	0.626	34	3.3	0.375	50	4.9	0.632
2	0.1	1.191	19	1.8	0.651	35	3.4	0.372	51	5.0	0.591
3	0.2	1.112	20	1.9	0.724	36	3.5	0.391	52	5.1	0.559
4	0.3	1.013	21	2.0	0.649	37	3.6	0.396	53	5.2	0.597
5	0.4	0.991	22	2.1	0.649	38	3.7	0.405	54	5.3	0.625
6	0.5	0.885	23	2.2	0.694	39	3.8	0.428	55	5.4	0.739
7	0.6	0.831	24	2.3	0.644	40	3.9	0.429	56	5.5	0.710
8	0.7	0.847	25	2.4	0.624	41	4.0	0.523	57	5.6	0.729
9	0.8	0.786	26	2.5	0.661	42	4.1	0.562	58	5.7	0.720
10	0.9	0.725	27	2.6	0.612	43	4.2	0.607	59	5.8	0.636
11	1.0	0.746	28	2.7	0.558	44	4.3	0.653	60	5.9	0.581
12	1.1	0.679	29	2.8	0.533	45	4.4	0.672	61	6.0	0.428
13	1.2	0.608	30	2.9	0.495	46	4.5	0.708	62	6.1	0.292
14	1.3	0.655	31	3.0	0.500	47	4.6	0.633	63	6.2	0.162
15	1.4	0.616	32	3.1	0.423	48	4.7	0.668	64	6.3	0.098
16	1.5	0.606	33	3.2	0.395	49	4.8	0.645	65	6.4	0.054
17	1.6	0.602									

Table 1.2.1: Caelli data

tions

$$\begin{aligned}
\phi_1(t) &= e^{-t} \\
\phi_2(t) &= e^{-(t-2)^2} \\
\phi_3(t) &= e^{-(t-2)^2} + e^{-(t-5)^2}
\end{aligned}$$

are plotted in Figure 1.2.7. Modelling the data by three Gaussians plus an exponential decay produces the approximating function (1.2.13)

$$\phi(\mathbf{x}; t) = x_1 e^{-x_5 t} + x_2 e^{-x_6 (t - x_9)^2} + x_3 e^{-x_7 (t - x_{10})^2} + x_4 e^{-x_8 (t - x_{11})^2}. \tag{1.2.13}$$

The standard initial guess at the value of the parameter vector $\mathbf{x}$ *is*

$$\mathbf{x}^{(1)} = [\ 1.3 \quad 0.65 \quad 0.65 \quad 0.7 \quad 0.6 \quad 3 \quad 5 \quad 7 \quad 2 \quad 4.5 \quad 5.5\ ]^T.$$

The most common measure of the difference between the approximating function (1.2.13) and the data values $y_i, i = 1, \ldots, m$ *is the sum of squares objective*

$$f(\mathbf{x}) = \sum_{i=1}^{m} |\phi(\mathbf{x}; t_i) - y_i|^2 = \|\mathbf{r}(\mathbf{x})\|_2^2. \tag{1.2.14}$$

For $i = 1, \ldots, m$

$$r_i(\mathbf{x}) = \phi(\mathbf{x}; t_i) - y_i$$

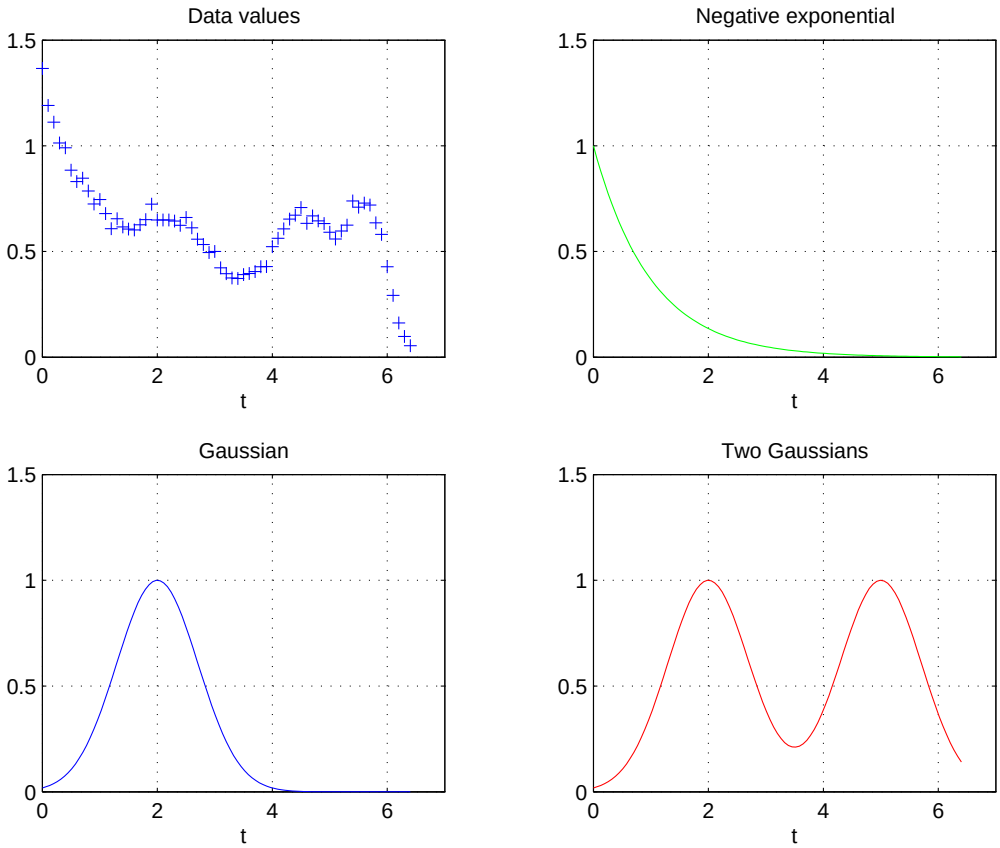


Figure 1.2.7: Caelli data and component functions

are the residuals, or differences between the approximating function $\phi(\mathbf{x}; t)$ *evaluated at* t_i *and the observed data values* y_i *at* t_i*. Minimizing (1.2.14) produces*

$$\mathbf{x}^* = \begin{bmatrix} 1.310363 \\ 0.4296501 \\ 0.6342121 \\ 0.6053728 \\ 0.7619573 \\ 0.8583250 \\ 1.405480 \\ 4.774411 \\ 2.405217 \\ 4.569355 \\ 5.673677 \end{bmatrix}.$$

The data, the initial approximating function $\phi(\mathbf{x}^{(1)}; t)$ *and the best least squares approximating function* $\phi(\mathbf{x}^*; t)$ *are plotted in Figure 1.2.8.*

In cases where there are large errors in the data measurements y_i *then minimizing the 1-norm*

$$\|\mathbf{r}(\mathbf{x})\|_1 = \sum_{i=1}^{m} |\phi(\mathbf{x}; t_i) - y_i)|$$

of the residual provides a robust solution. On the other hand if the data y_i *is known exactly, for example from sampling a function at specified points, then minimizing the* ∞*-norm*

$$\|\mathbf{r}(\mathbf{x})\|_\infty = \max_{i=1,\ldots,m} |\phi(\mathbf{x}; t_i) - y_i)|$$

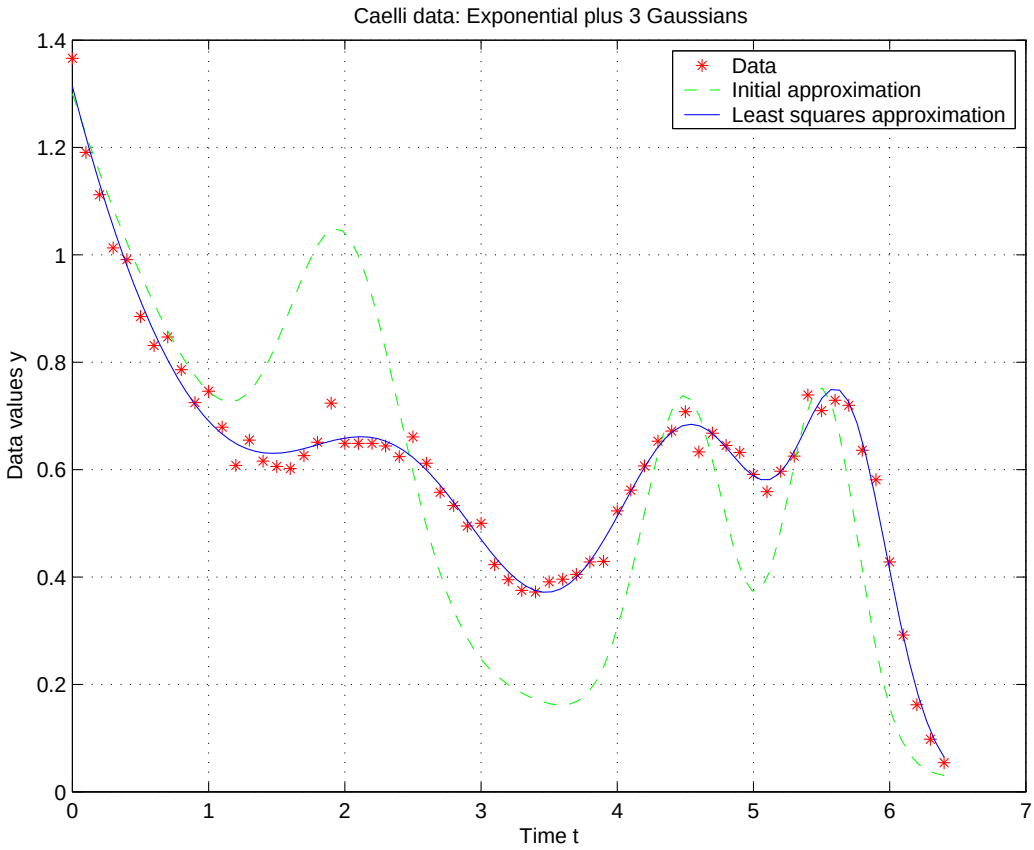


Figure 1.2.8: Caelli data and approximating functions

provides a solution which bounds the magnitude of the worst error at the data points. Data fitting and function approximation are discussed further in Chapter 9.

1.2.8 Logic

Already the terms "sufficient condition" and "necessary condition" have been used when talking about conditions for existence of global extrema or for a point to be a local minimizer. A clear understanding of what these terms mean, and the basics of mathematical logic, is essential.

Let P and Q represent two statements.

1. Q is a *necessary condition* for P if the statement P being true implies that the statement Q must be true, i.e. $\mathrm{P} \Longrightarrow \mathrm{Q}$.

2. Q is a *sufficient condition* for P if the statement Q being true implies that the statement P must be true, i.e. $\mathrm{Q} \Longrightarrow \mathrm{P}$.

3. Q is a *necessary and sufficient condition* for P if the statement Q is true if and only if the statement P is true, i.e. $\mathrm{Q} \iff \mathrm{P}$.

If $\mathrm{P} \Longrightarrow \mathrm{Q}$ then the *converse*, $\mathrm{Q} \Longrightarrow \mathrm{P}$, may or may not be true. However if $\mathrm{P} \Longrightarrow \mathrm{Q}$ then the *contrapositive*, not Q $\Longrightarrow$ not P, is automatically true.

Example 1.2.12 (Logic) *Fred is taking the subject MATH0001, The Use of Mathematics. Two statements on his performance in this subject are*

- *P = "Fred's mark in MATH0001 is at least 50"*

- *Q = "Fred passed MATH0001".*

Then $P \Longrightarrow Q$, so P is a sufficient condition for Q. The contrapositive, not $Q \Longrightarrow$ not P, is "Fred failed MATH0001" so "Fred's mark in MATH0001 is less than 50".

The statements P and Q are not necessary and sufficient as it is possible to get a conceded pass on a mark of 49 if Fred's grades in other subjects are good enough. Fred cannot get any form of a pass if his mark is below 47. Thus a necessary condition for Q is "Fred's mark in MATH0001 is at least 47". This is not a sufficient condition for Q as Fred's result with marks of 49, 48 or 47 depends on his results in other subjects.

The ideal situations is to have necessary and sufficient conditions for identifying a (local) minimizer. However unless the objective function has some special properties this is rarely possible. Necessary conditions for a local minimizer are of the form "if $\mathbf{x}^*$ is a local minimizer of $f(\mathbf{x})$ then P holds". The contrapositive "if P does not hold then $\mathbf{x}^*$ is not a local minimizer" means that a search for local minimizers need only consider points for which the statement P is true. However it does not mean that every point for which P is true is a local minimizer. A sufficient condition Q for a local minimizer is of the form "if Q is true then $\mathbf{x}^*$ is a local minimizer of $f(\mathbf{x})$".

Typically necessary conditions and sufficient conditions for local minima involve the first and second derivatives of the objective function and the constraint functions.

1.2.9 Exercises

1. Discuss the existence of global extrema (both minima and maxima) of the following problems.
   (a) $f(x) = e^{-(x-1)^2}$ on $\Omega = [0, \infty)$.
   (b) $f(\mathbf{x}) = \cos\left(\frac{x_1^2+x_2^2}{5}\right)$ on $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1 + x_2 \leq \pi, x_1 \geq 0, x_2 \geq -\pi\}$.
   (c) $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 2)^2$ on $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1 \geq 0, x_2 \geq 0\}$.
2. Use `Maple` to
   (a) i. Define the function
   $$f(\mathbf{x}) = x_1 e^{-x_1^2 - x_2^2}.$$
   ii. Draw a 3-D plot of the function over the region
   $$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : |x_1| \leq 2, |x_2| \leq 2\}.$$
   Use the left mouse button to rotate the plot and the middle mouse button to re-draw the plot. Also try boxed axes and different plot styles.
   iii. Calculate the gradient $\nabla f(\mathbf{x})$ and Hessian $\nabla^2 f(\mathbf{x})$.
   A `Maple` text file is available from the ocurse web page in the file `fplt.txt`.
   iv. Repeat this question for
   $$f(\mathbf{x}) = \sin(\pi(x_1 + x_2^2))e^{-x_1^2 - x_2^2}.$$
   Also write a MATLAB file to draw a 3-D mesh plot and contour plot of $f(\mathbf{x})$ on Ω.
   A `Maple` text file is available in the file `mbq2a.txt` and a MATLAB file is available in `mbq2a.m`.
   (b) Use `Maple` to
   i. Define the quantities
   $$G = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \mathbf{g}_0 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad f_0 = 1.$$
   ii. Define the quadratic function
   $$q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G \mathbf{x} + \mathbf{x}^T \mathbf{g}_0 + f_0.$$
   Remember that `Maple` uses `&*` to signify non-commutative matrix multiplication and the command `evalm` is needed to evaluate matrix-valued expressions.

iii. Calculate the gradient $\nabla q(\mathbf{x})$ and Hessian $\nabla^2 q(\mathbf{x})$.

iv. Calculate the eigenvalues and eigenvectors of G.

v. Draw a 3-D plot of the quadratic function, noting how the function increases in the directions of the eigenvectors.

vi. Change the matrix G to

$$\begin{bmatrix} -2 & -1 \\ -1 & 2 \end{bmatrix}$$

and repeat the calculations.

vii. Change the matrix G to

$$\begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix}$$

and repeat the calculations.

A `Maple` text file is available in the file `quads.txt` in the class account.

3. Let $f(\mathbf{x}) = 3(x_1^2 - 2x_2x_3)^2 - 6x_1e^{x_2^2}$ and consider the points $\hat{\mathbf{x}} = [\ 2 \quad 2 \quad -1\ ]^T$ and $\bar{\mathbf{x}} = [\ -1 \quad 2 \quad 3\ ]^T$.

   (a) Calculate the gradient $\nabla f(\mathbf{x})$ and the Hessian $\nabla^2 f(\mathbf{x})$ of $f(\mathbf{x})$.

   (b) Find the parametric form $\mathbf{x}(\alpha) = \bar{\mathbf{x}} + \alpha\mathbf{d}$, where $\|\mathbf{d}\| = 1$, of the line through $\hat{\mathbf{x}}$ and $\bar{\mathbf{x}}$.

   (c) Let $\ell(\alpha) \equiv f(\mathbf{x}(\alpha))$. Show that

$$\ell(\alpha) = \frac{243}{625}\alpha^4 + \frac{108}{25}\alpha^3 - \frac{294}{25}\alpha^2 + \left(-\frac{18}{5}e^4 - 132\right)\alpha + 363 + 6e^4.$$

   and hence find $\ell'(\alpha)$ and $\ell''(\alpha)$.

   (d) At the point $\tilde{\mathbf{x}} = [\ 0 \quad 2 \quad 5/3\ ]^T$ verify that $\ell'(\alpha) = \mathbf{d}^T\nabla f(\bar{\mathbf{x}} + \alpha\mathbf{d})$ and $\ell''(\alpha) = \mathbf{d}^T\nabla^2 f(\bar{\mathbf{x}} + \alpha\mathbf{d})\mathbf{d}$.

   (e) Check your answers using `Maple`.

4. Determine whether the following matrices are positive definite, positive semi-definite, negative definite, negative semi-definite or indefinite.

   (a)

$$\text{(i)} \begin{bmatrix} 3 & -5 \\ -5 & 6 \end{bmatrix} \quad \text{(ii)} \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \quad \text{(iii)} \begin{bmatrix} -3 & -2 \\ -2 & -6 \end{bmatrix} \quad \text{(iv)} \begin{bmatrix} 5 & -3 \\ -3 & 3 \end{bmatrix}$$

   (b)

$$\text{(i)} \begin{bmatrix} 2 & 0 & -5 \\ 0 & 3 & 0 \\ -5 & 0 & 14 \end{bmatrix} \quad \text{(ii)} \begin{bmatrix} -2 & 3 & 0 \\ 3 & -3 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad \text{(iii)} \begin{bmatrix} -2 & 2 & 0 \\ 2 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

   (c) Check your answers by using `Maple` or MATLAB to find the eigenvalues of these matrices.

5. Let $G \in \mathbb{R}^{n\times n}$ be a symmetric matrix with a factorization $G = LDL^T$, where L is a unit lower triangular matrix and D is a diagonal matrix.

   (a) Show that G is positive definite if and only if $D_{ii} > 0$ for $i = 1, \ldots, n$.

   (b) Calculate the LDL^T factorization of G, given

$$G = \begin{bmatrix} 2 & 4 & -6 \\ 4 & 9 & -13 \\ -6 & -13 & 22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -6 \\ 0 & 1 & -1 \\ 0 & 0 & 3 \end{bmatrix}$$

   and show that G is positive definite.

   (c) Calculate the Cholesky factorization of $G = \hat{L}\hat{L}^T$ for the matrix G in the previous part.

6. Let A be an n by n positive definite matrix. Prove that $f(\mathbf{x}) = (\mathbf{x}^T A\mathbf{x})^{\frac{1}{2}}$ is a norm.

   Hint: For the triangle inequality use the Cholesky factorization (see the Appendix of matrix algebra) and the triangle inequality for 2-norm.

7. Prove that an $n \times n$ symmetric matrix G is positive definite if and only if G^{-1} is positive definite.

8. Let $q\colon \mathbb{R}^n \to \mathbb{R}$ be a quadratic function.

   (a) Show that the Hessian of q maps differences in points into differences in gradients, that is

   $$G\,(\bar{\mathbf{x}} - \hat{\mathbf{x}}) = \bar{\mathbf{g}} - \hat{\mathbf{g}},$$

   where $\bar{\mathbf{g}} = \nabla q(\bar{\mathbf{x}})$, $\hat{\mathbf{g}} = \nabla q(\hat{\mathbf{x}})$ and $G = \nabla^2 q(\mathbf{x})$.

   (b) When q is a positive definite quadratic function show how an evaluation of $q(\mathbf{x})$, $\nabla q(\mathbf{x})$ and $\nabla^2 q(\mathbf{x})$ at a single point allows you to calculate the minimizer $\mathbf{x}^*$ of q and the minimum $q(\mathbf{x}^*)$.

1.3 Problem Structure

In standard form our optimization problem is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & f(\mathbf{x}) \\ Subject\ to & c_i(\mathbf{x}) = 0 \text{ for } i \in \mathcal{E} \\ & c_i(\mathbf{x}) \leq 0 \text{ for } i \in \mathcal{I}. \end{array}$$

To obtain efficient methods we must make full use of the available information (e.g. function values, gradients, and Hessians) and the *structure* of the problem.

1.3.1 Number of variables n

Here there are only two important cases:

1. One-dimensional problem: $n = 1$.
2. Multi-variable problem: $n > 1$.

For multi-variable problems the number of variables is one measure of the size of the problem, and larger problems require more work to find a local minimizer.

1.3.2 Objective function $f(\mathbf{x})$

Firstly assume that the objective function $f \in C^2$, so that it is smooth. When a function is nonsmooth it is important to understand exactly how the nonsmooth behaviour arises, and to use that to produce efficient methods.

1. No objective:

   The problem is to find a *feasible point* $\mathbf{x} \in \Omega$, that is a point $\mathbf{x} \in \mathbb{R}^n$ satisfying *all* the constraints $c_i(\mathbf{x}) = 0$ $i \in \mathcal{E}$ and $c_i(\mathbf{x}) \leq 0$ $i \in \mathcal{I}$.

2. Affine: $f(\mathbf{x}) = \mathbf{g}^T\mathbf{x} + f_0$

   For an affine function the gradient $\nabla f(\mathbf{x}) \equiv \mathbf{g}(\mathbf{x}) = \mathbf{g}$ does not depend on the variables $\mathbf{x}$, and the Hessian $\nabla^2 f(\mathbf{x}) \equiv G(\mathbf{x})$ is the n by n zero matrix.

3. Quadratic: $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{g}_0^T\mathbf{x} + f_0$

   For a quadratic function the gradient $\nabla f(\mathbf{x}) \equiv \mathbf{g}(\mathbf{x}) = G\mathbf{x} + \mathbf{g}_0$ is a affine function of $\mathbf{x}$, and the Hessian $\nabla^2 f(\mathbf{x}) \equiv G(\mathbf{x}) = G$ is an n by n symmetric matrix which does not depend on the variables $\mathbf{x}$. The case when G is positive definite, so f is a strictly convex quadratic function, is important as this is the simplest function with a unique unconstrained minimizer.

4. General nonlinear: $f(\mathbf{x})$

   In this case both the gradient $\nabla f(\mathbf{x}) \equiv \mathbf{g}(x)$ and the Hessian $\nabla^2 f(\mathbf{x}) \equiv G(x)$ are functions of $\mathbf{x}$.

5. Sum of squares: $f(\mathbf{x}) = \sum_{i=1}^{m} [r_i(\mathbf{x})]^2 = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$

   Here $\mathbf{r}(\mathbf{x}) \equiv (\ r_1(\mathbf{x}), r_2(\mathbf{x}), \ldots, r_m(\mathbf{x})\ )^T$ is an m-dimensional vector valued function of the variables $\mathbf{x}$. The m by n *Jacobian* matrix $J(\mathbf{x})$ is defined by

   $$J_{ij}(\mathbf{x}) = \frac{\partial r_i(x)}{\partial x_j} \qquad \text{for } i = 1, \ldots, m\ ,\ j = 1, ..., n. \tag{1.3.1}$$

   Then the gradient and Hessian are

   $$\nabla f(\mathbf{x}) = 2J(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \qquad \nabla^2 f(\mathbf{x}) = 2J(\mathbf{x})^T J(\mathbf{x}) + 2\sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}) \tag{1.3.2}$$

   If $J(\mathbf{x})$ is independent of $\mathbf{x}$, so the functions $r_i(\mathbf{x})$ are affine, $\mathbf{r}(\mathbf{x}) = J\mathbf{x} + \mathbf{r}_0$, and f is a special quadratic function.

   Minimizing a sum of squares frequently arises in connection with solving systems of nonlinear equations $\mathbf{r}(\mathbf{x}) = 0$, and data fitting problems.

6. Separable: $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$.

   Here $f(\mathbf{x})$ is a sum of n independent functions f_i each of which is a function of a single variable x_i. This problem is separable into n one-dimensional problems.

7. Partially Separable: $f(\mathbf{x}) = \sum_{\ell=1}^{L} f_\ell(\mathbf{x}_\ell)$

   Here each function $f_\ell(\mathbf{x})$ depends on only a subset of the variables $\mathbf{x}_\ell$ and the subsets of variables are disjoint. The key feature is that the variables $\mathbf{x}_\ell$ on which $f_\ell(\mathbf{x})$ depends do NOT appear in the definition of any of the other component functions. An example with $n = 10, L = 3$ is

   $$\begin{aligned} f_1(\mathbf{x}) &= f_1(x_2, x_3, x_5, x_6, x_9, x_{10}) \\ f_2(\mathbf{x}) &= f_2(x_1, x_4, x_8) \\ f_3(\mathbf{x}) &= f_3(x_7) \end{aligned}$$

   so $\mathbf{x} = [\ \mathbf{x}_1^T \quad \mathbf{x}_2^T \quad \mathbf{x}_3^T\ ]^T$ where $\mathbf{x}_1 = [\ x_2 \quad x_3 \quad x_5 \quad x_6 \quad x_9 \quad x_{10}\ ]^T$, $\mathbf{x}_2 = [\ x_1 \quad x_4 \quad x_8\ ]^T$, $\mathbf{x}_3 = [\ x_7\ ]^T$.

8. Multi-objective problems: $f_i(\mathbf{x})$ for $i = 1, \ldots, \ell$

   There are $\ell > 1$ objective functions, all of which should be minimized at the same time.

9. Minimax problems: $f(\mathbf{x}) = \max_{i=1,\ldots,m} f_i(\mathbf{x})$

   Here, even though each component function $f_i(\mathbf{x})$ is typically smooth, the objective function is generally not smooth at points where more than one of the component functions $f_i(\mathbf{x})$ achieve the maximum.

10. Stochastic problems: $f(\mathbf{x}) = E[\psi(\mathbf{x}; \sigma)]$

    The objective value is the expected value of a function which depends on one or more stochastic variables σ. If σ has a discrete distribution with probability density function $P(\sigma = sigma_i) = p_i$ for $i = 1, \ldots, m$ then the expected value is

    $$E[\psi(\mathbf{x}; \sigma)] = \sum_{i=1}^{m} \psi(\mathbf{x}; \sigma_i) p_i.$$

    If σ has a continuous distribution with probability density function $p(\sigma)$ then the expected value is

    $$\mathcal{E}[\,\psi(\mathbf{x}; \sigma)\,] = \int_{-\infty}^{\infty} \psi(\mathbf{x}; \sigma) p(\sigma) d\sigma.$$

1.3.3 Specifying the feasible region Ω

Most commonly the feasible region is specified by systems of equalities and inequalities, so

$$\Omega = \{\, \mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \text{ for } i \in \mathcal{E}, c_i(\mathbf{x}) \leq 0 \text{ for } i \in \mathcal{I} \,\}.$$

Here it is typically assumed the functions $c_i(\mathbf{x})$ are twice continuously differentiable.

Commonly occurring cases are

1. Unconstrained problem: $\mathcal{E} = \emptyset$ and $\mathcal{I} = \emptyset$.

   In this case there are no restrictions on the variables.

2. Simple lower and upper bounds: $l_i \leq x_i \leq u_i$ for $i = 1, \ldots, n$.

3. Network constraints: Conservation of flow in a network implies that the total flow into a node must equal the total flow out of a node. Network constraints frequently involve the node-arc incidence matrix. Let the nodes i be indexed $\{1, ..., m\}$ and the arcs j indexed $\{1, ..., n\}$, and let

$$A_{ij} = \begin{cases} +1 & \text{if arc } j \text{ leaves node } i; \\ -1 & \text{if arc } j \text{ enters node } i; \\ 0 & \text{otherwise} \end{cases} \tag{1.3.3}$$

   Conservation of flow leads to the system of linear equations $A\mathbf{x} = \mathbf{b}$, where x_j represent the flow on arc j, and b_i is the external flow at each node i.

4. General linearly constrained problem: $c_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - \beta_i$

   A linear constraint has gradient $\nabla c_i(\mathbf{x}) = \mathbf{a}_i$ which is independent of $\mathbf{x}$, and its Hessian matrix $\nabla^2 c_i(\mathbf{x})$ is the n by n zero matrix.

   Simple lower or upper bounds are just a special case where $\mathbf{a}_j = \pm\mathbf{e}_i$ (where $\mathbf{e}_i$ is the i th unit vector in $\mathbb{R}^n$).

5. Quadratic constraints: $c(\mathbf{x}) = \mathbf{x}^T C\mathbf{x} + \mathbf{a}^T x + \delta$

   Generally quadratic constraints are as hard as general nonlinear constraints. One special case is when $\delta > 0, \mathbf{a} = 0$ and $C = -I$, so

$$c(\mathbf{x}) = -\mathbf{x}^T\mathbf{x} + \delta \geq 0 \quad \Longleftrightarrow \quad ||\mathbf{x}||_2 \leq \delta^{\frac{1}{2}}.$$

6. Nonlinear constraints:

   Unfortunately the problem becomes considerably more difficult if any nonlinear constraints (even quadratic constraints) are present.

7. Discrete constraints: $x_i \in \mathcal{D}_i$ for $i \in N$

   N is a subset of $\{1, \ldots, n\}$ specifying the variables which must take values in the discrete sets $\mathcal{D}_i$. Typical examples are integrality constraints where x_i must be a integer for $i \in N$, or zero-one variables where $\mathcal{D} = \{0, 1\}$. The presence of integrality constraints often means that some variables can only take a finite number of values. This introduces a combinatorial aspect to the problem which increases, not decreases, the difficulty of the problem.

8. Semi-infinite constraints: $c(\mathbf{x}; t) \leq 0$ for all $t \in T$

   Instead of having a finite set of constraints $c_i(\mathbf{x}) \leq 0$ for $i \in \mathcal{I}$, the constraints must be satisfied for all values of t in a set $T \subseteq \mathbb{R}^s$. Often T is an interval ($s = 1$). The name semi-infinite comes from the fact that there are a finite number of variables $\mathbf{x} \in \mathbb{R}^n$, but an infinite number of constraints for $t \in T$.

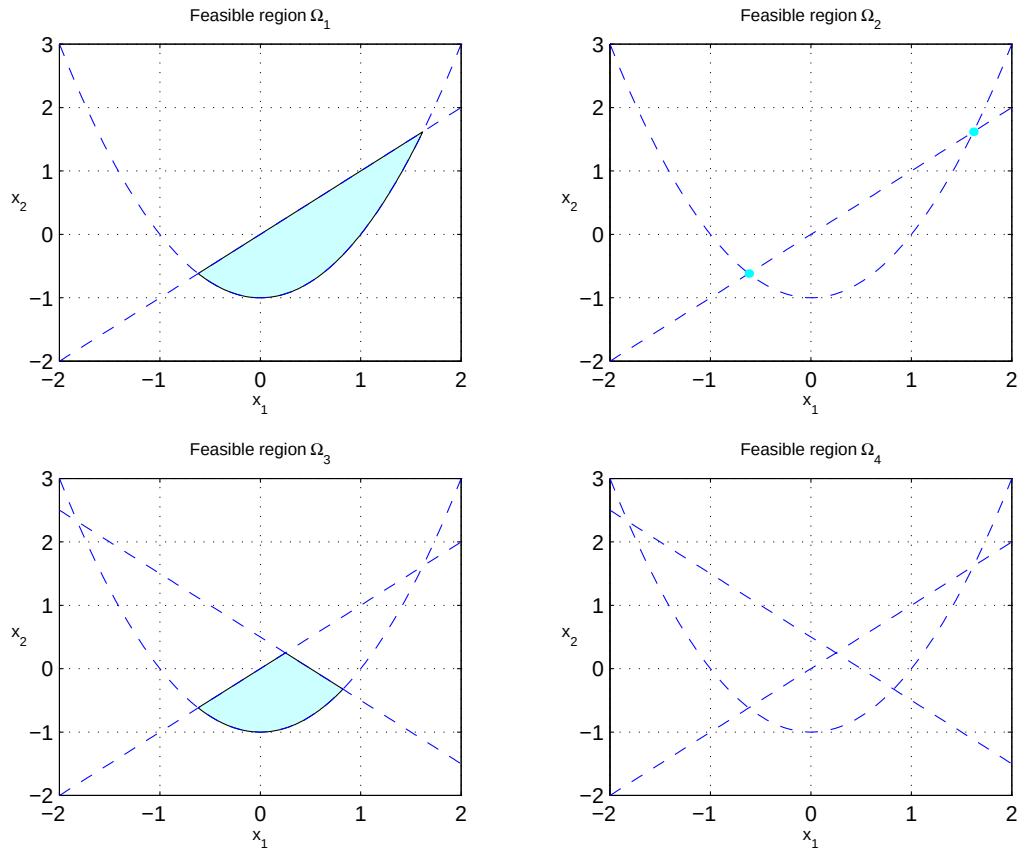


Figure 1.3.1: Feasible regions $\Omega_1, \ldots, \Omega_4$

Problems can arise in which the feasible region is empty because of inconsistent constraints, the feasible region just consists of isolated points or there are redundant constraints. Consider the feasible region

$$\Omega = \{ \mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0, i \in \mathcal{E},\ c_i(\mathbf{x}) \leq 0, i \in \mathcal{I} \},$$

and the functions

$$\begin{aligned}
c_1(\mathbf{x}) &= x_1^2 - x_2 - 1 \\
c_2(\mathbf{x}) &= -x_1 + x_2 \\
c_3(\mathbf{x}) &= x_1 + x_2 - \tfrac{1}{2} \\
c_4(\mathbf{x}) &= x_1 - x_2 + 1 \\
c_5(\mathbf{x}) &= -4x_1 + 4x_2 - 4 \\
c_6(\mathbf{x}) &= -4x_1 + 4x_2 + 4
\end{aligned}$$

The four plots in Figure 1.3.1 sketch the constraint functions defining the sets corresponding to $\mathcal{E}_1 = \emptyset, \mathcal{I}_1 = \{1, 2\}, \mathcal{E}_2 = \{1, 2\}, \mathcal{I}_2 = \emptyset, \mathcal{E}_3 = \emptyset, \mathcal{I}_3 = \{1, 2, 3\}, \mathcal{E}_4 = \{1, 2, 3\}, \mathcal{I}_4 = \emptyset$.

In Figure 1.3.1 Ω_1, defined by inequality constraints, consists of a compact connected region in $\mathbb{R}^2$. On the other hand Ω_2 is defined by the same functions, but as equality constraints, giving

$$\begin{aligned}
\Omega_2 &= \{ \mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) = 0, c_2(\mathbf{x}) = 0 \} \\
&= \left\{ \frac{1+\sqrt{5}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \frac{1-\sqrt{5}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.
\end{aligned}$$

which consists of the two distinct points near $-0.618\mathbf{e}$ and $1.618\mathbf{e}$. Generally if there are $m = n$ equality constraints defining the feasible region Ω then the feasible region consists of isolated points in $\mathbb{R}^n$.

The sets Ω_3 and Ω_4 are defined by the same functions, but Ω_3 has inequality constraints, while Ω_4 has equality constraints. In particular

$$\Omega_4 = \left\{ \mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) = 0, c_2(\mathbf{x}) = 0, c_3(\mathbf{x}) = 0 \right\} = \emptyset.$$

There are no points in $\mathbb{R}^2$ which satisfy *all* the equality constraints, so the feasible region is empty. In general if there are $m > n$ equality constraints then the feasible region will be empty. This situation can arise because the constraints in the mathematical model are too stringent, or an error has been made in formulating the mathematical model. In either case the underlying physical problem should be carefully examined.

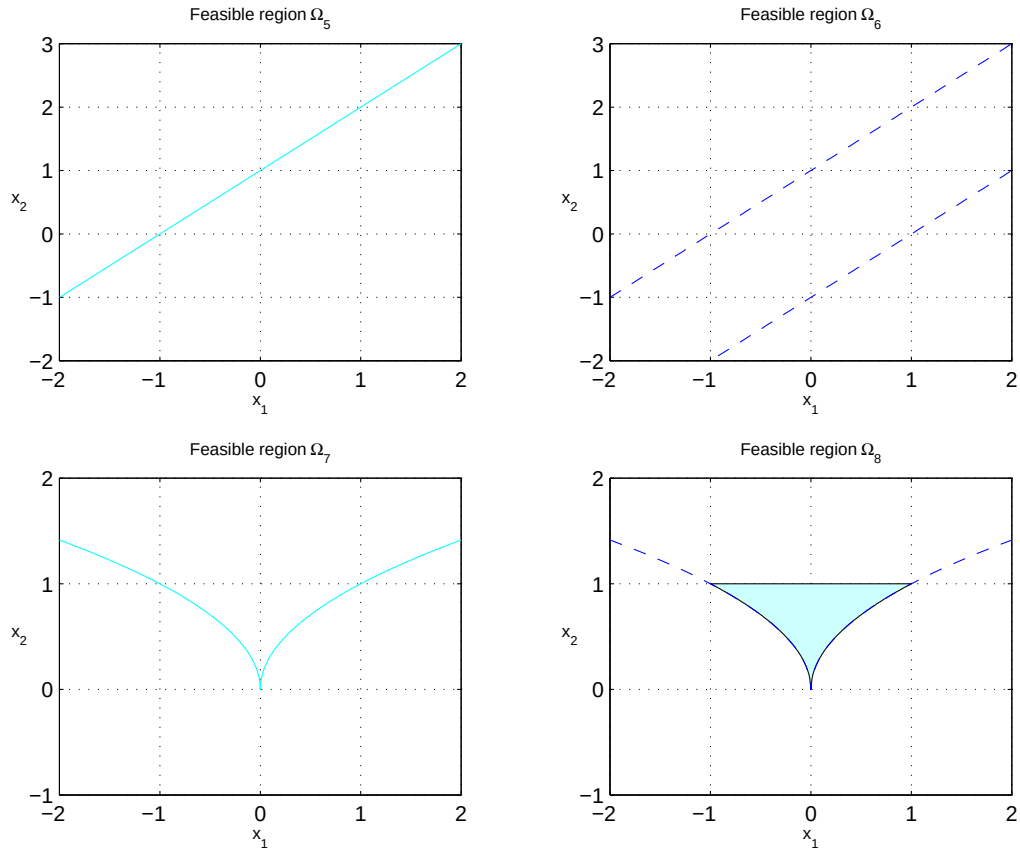


Figure 1.3.2: Feasible regions $\Omega_5, \ldots, \Omega_8$

In Figure 1.3.2

$$\begin{aligned}
\Omega_5 &= \left\{ \mathbf{x} \in \mathbb{R}^2 : c_4(\mathbf{x}) = 0, c_5(\mathbf{x}) = 0 \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 + 1 = 0, -4x_1 + 4x_2 - 4 = 0 \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{R}^2 : x_2 = x_1 + 1 \right\} \\
&= \left\{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = \begin{bmatrix} \alpha \\ \alpha + 1 \end{bmatrix}, \alpha \in \mathbb{R} \right\}.
\end{aligned}$$

The feasible region Ω_5 is a line in $\mathbb{R}^2$. However only one of the two constraints is required to specify the feasible region, and the other constraint is redundant. This is another situation that should be avoided when formulating the mathematical model.

$$\Omega_6 = \left\{ \mathbf{x} \in \mathbb{R}^2 : c_4(\mathbf{x}) = 0, c_6(\mathbf{x}) = 0 \right\} = \left\{ \mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 + 1 = 0, -4x_1 + 4x_2 + 4 = 0 \right\} = \emptyset.$$

The feasible region is empty as the two constraints cannot be satisfied simultaneously. Thus even if the number of equality constraints $m \leq n$ the number of variables, the feasible region may be empty. The only difference between the functions specifying Ω_5 and Ω_6 is the constant term in $c_5(\mathbf{x})$ and $c_6(\mathbf{x})$. Again this would point to possible errors in the formulation of the mathematical model.

When there are nonlinear constraints the problem of finding a feasible point can be a difficult problem in its own right. Typically optimality conditions and algorithms require some form of regularity condition on the geometry of the feasible region or on the algebraic description. For example let

$$c_7(\mathbf{x}) \equiv \sqrt{|x_1|} - x_2.$$

Part of the feasible region

$$\Omega_7 = \left\{ \mathbf{x} \in \mathbb{R}^2 : c_7(\mathbf{x}) = 0 \right\}$$

is sketched in Figure 1.3.2. The constraint function $c_1(\mathbf{x})$ is not differentiable at $\bar{\mathbf{x}} = 0$, which will cause difficulties in developing both optimality conditions and algorithms. Finally

$$\Omega_8 = \left\{ \mathbf{x} \in \mathbb{R}^2 : c_7(\mathbf{x}) \leq 0, x_2 \leq 1 \right\}$$

has a boundary which is nonsmooth. If the solution is not close to the origin then this may not cause a problem. However difficulties are likely to occur close to the origin.

1.3.4 Available information

When the objective function is linear and all the constraints are linear the problem is referred to as a *linear programming* problem. When the objective function is quadratic and all the constraints are linear the problem is referred to as a *quadratic programming* problem. There are many excellent books on applications and techniques for linear programming, as it is the most widely used optimization technique.

For many large problems the Hessian, Jacobian or the constraint gradients have a large number of elements which are known to be zero. If the number of non-zero elements is a small fraction of the total number of elements then the problem is *sparse*. Efficient methods must make full use of this sparsity structure, both to save storage and to speed up matrix calculations. Example A.2.1 on page 226 of Appendix A gives an example of a sparse banded matrix arising from the discretization of a partial differential equation.

As well as the structure of the problem the amount of information that is available and can be readily calculated is important. Typically algorithms require the user to specify

1. function values only, or
2. function and gradient values, or
3. function, gradient and Hessian values

of the objective and constraint functions. Generally if more information is available then more efficient and robust algorithms are available to solve that optimization problem. Moreover on large problems more efficient numerical methods can be developed by exploiting the structure of the problem. This includes separability of the functions, sparsity of Hessian and constraint matrices, and simple bounds, as opposed to general linear constraints, on the variables.

Example 1.3.1 (Partially separable functions) *The function used in Example 1.2.11 to approximate the Caelli data is*

$$\phi(\mathbf{x}; t) = x_1 e^{-x_5 t} + x_2 e^{-x_6 (t - x_9)^2} + x_3 e^{-x_7 (t - x_{10})^2} + x_4 e^{-x_8 (t - x_{11})^2}.$$

The variables are $\mathbf{x} \in \mathbb{R}^{11}$ *and* $t \in \mathbb{R}$ *is a parameter.*

1. *Find the gradient and Hessian of* $f(\mathbf{x}; t)$. *Check your answers using* `Maple`.
2. *Calculate the sparsity of the Hessian of* $f(\mathbf{x}; t)$.

3. *Show that $f(\mathbf{x};t)$ is a partially separable function as it can be written as*

$$f(\mathbf{x};t) = \sum_{i=1}^{4} f_i(\mathbf{x};t)$$

where each function $f_i(\mathbf{x};t)$ involves at most 3 of the variables x_i.

4. *Re-order the variables to produce a partially separable function $h(\mathbf{y};t)$ which is equivalent to $f(\mathbf{x};t)$, but the Hessian of $h(\mathbf{y};t)$ is block diagonal.*

The gradient and Hessian of $\phi(\mathbf{x};t)$ with respect to the $\mathbf{x}$ variables are

$$\nabla_{\mathbf{x}}\phi(\mathbf{x};t) = \begin{bmatrix} e^{-x_5 t} \\ e^{-x_6(t-x_9)^2} \\ e^{-x_7(t-x_{10})^2} \\ e^{-x_8(t-x_{11})^2} \\ -x_1 t e^{-x_5 t} \\ -x_2(t-x_9)^2 e^{-x_6(t-x_9)^2} \\ -x_3(t-x_{10})^2 e^{-x_7(t-x_{10})^2} \\ -x_4(t-x_{11})^2 e^{-x_8(t-x_{11})^2} \\ 2x_2x_6(t-x_9)e^{-x_6(t-x_9)^2} \\ 2x_3x_7(t-x_{10})e^{-x_7(t-x_{10})^2} \\ 2x_4x_8(t-x_{11})e^{-x_8(t-x_{11})^2} \end{bmatrix}$$

and

$$\nabla^2_{\mathbf{x}}\phi(\mathbf{x};t) = \begin{bmatrix} 0 & 0 & 0 & 0 & G_{1,5} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & G_{2,6} & 0 & 0 & G_{2,9} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & G_{3,7} & 0 & 0 & G_{3,10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & G_{4,8} & 0 & 0 & G_{4,11} \\ G_{1,5} & 0 & 0 & 0 & G_{5,5} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & G_{2,6} & 0 & 0 & 0 & G_{6,6} & 0 & 0 & G_{6,9} & 0 & 0 \\ 0 & 0 & G_{3,7} & 0 & 0 & 0 & G_{7,7} & 0 & 0 & G_{7,10} & 0 \\ 0 & 0 & 0 & G_{4,8} & 0 & 0 & 0 & G_{8,8} & 0 & 0 & G_{8,11} \\ 0 & G_{2,9} & 0 & 0 & 0 & G_{6,9} & 0 & 0 & G_{9,9} & 0 & 0 \\ 0 & 0 & G_{3,10} & 0 & 0 & 0 & G_{7,10} & 0 & 0 & G_{10,10} & 0 \\ 0 & 0 & 0 & G_{4,11} & 0 & 0 & 0 & G_{8,11} & 0 & 0 & G_{11,11} \end{bmatrix}$$

where

$$\begin{aligned}
G_{1,5} &= -te^{-x_5 t}, \\
G_{2,6} &= -(t-x_9)^2 e^{-x_6(t-x_9)^2}, \\
G_{2,9} &= 2x_6(t-x_9)e^{-x_6(t-x_9)^2}, \\
G_{3,7} &= -(t-x_{10})^2 e^{-x_7(t-x_{10})^2}, \\
G_{3,10} &= 2x_7(t-x_{10})e^{-x_7(t-x_{10})^2}, \\
G_{4,8} &= -(t-x_{11})^2 e^{-x_8(t-x_{11})^2}, \\
G_{4,11} &= 2x_8(t-x_{11})e^{-x_8(t-x_{11})^2}, \\
G_{5,5} &= x_1 t^2, \\
G_{6,6} &= x_2(t-x_9)^4 e^{-x_6(t-x_9)^2}, \\
G_{6,9} &= -2x_2x_6 e^{-x_6(t-x_9)^2} + 4x_2x_6^2(t-x_9)^2 e^{-x_6(t-x_9)^2}, \\
G_{7,7} &= x_3(t-x_{10})^4 e^{-x_7(t-x_{10})^2},
\end{aligned}$$

$$
\begin{aligned}
G_{7,10} &= 2x_3(t-x_{10})e^{-x_7(t-x_{10})^2} - 2x_3x_7(t-x_{10})^3e^{-x_7(t-x_{10})^2}, \\
G_{8,8} &= x_4(t-x_{11})^4e^{-x_8(t-x_{11})^2}, \\
G_{8,11} &= 2x_4(t-x_{11})e^{-x_8(t-x_{11})^2} - 2x_4x_8(t-x_{11})^3e^{-x_8(t-x_{11})^2}, \\
G_{9,9} &= -2x_2x_6e^{-x_6(t-x_9)^2} + 4x_2x_6^2(t-x_9)^2e^{-x_6(t-x_9)^2}, \\
G_{10,10} &= -2x_3x_7e^{-x_8(t-x_{11})^2} + 4x_3x_7^2(t-x_{10})^2e^{-x_8(t-x_{11})^2}, \\
G_{11,11} &= -2x_4x_8e^{-x_8(t-x_{11})^2} + 4x_4x_8^2(t-x_{11})^2e^{-x_8(t-x_{11})^2}
\end{aligned}
$$

This is an example of a sparse matrix in which only 27 of the 121 elements are non-zero, corresponding to a sparsity of approximately 22%. The `Maple` function `sparsematrixplot` in the `plots` package or the MATLAB function `spy` can be used to plot the non-zero elements of a matrix, giving a good idea of the sparsity pattern. The function $\phi(\mathbf{x};t)$ is also an example of a partially separable function as

$$\phi(\mathbf{x};t) = \sum_{i=1}^{4} f_i(\mathbf{x};t),$$

where

$$f_1(\mathbf{x};t) = x_1e^{-x_5t}$$

depends only on x_1, x_5,

$$f_2(\mathbf{x};t) = x_2e^{-x_6(t-x_9)^2},$$

depends only on x_2, x_6 and x_9,

$$f_3(\mathbf{x};t) = x_3e^{-x_7(t-x_{10})^2}$$

depends only on x_3, x_7 and x_{10} and

$$f_4(\mathbf{x};t) = x_4e^{-x_8(t-x_{11})^2}$$

depends only on x_4, x_8 and x_{11}.

Re-ordering the variables so

$$\psi(\mathbf{y};t) = y_1e^{-y_2t} + y_3e^{-y_4(t-y_5)^2} + y_6e^{-y_7(t-y_8)^2} + y_9e^{-y_{10}(t-y_{11})^2}$$

produces a Hessian with the structure

$$
\nabla^2_{\mathbf{y}}\psi(\mathbf{y};t) = \begin{bmatrix}
0 & H_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H_{1,2} & H_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & H_{3,4} & H_{3,5} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & H_{3,4} & H_{4,4} & H_{4,5} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & H_{3,5} & H_{4,5} & H_{5,5} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & H_{6,7} & H_{6,8} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & H_{6,7} & H_{7,7} & H_{7,8} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & H_{6,8} & H_{7,8} & H_{8,8} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & H_{9,10} & H_{9,11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & H_{9,10} & H_{10,10} & H_{10,11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & H_{9,11} & H_{10,11} & H_{11,11}
\end{bmatrix}
$$

This matrix is block-diagonal, with a 2×2 block, then three 3×3 blocks. Setting

$$
\begin{aligned}
h_1(\mathbf{y}_1;t) &= y_1e^{-y_2t} \\
h_2(\mathbf{y}_2;t) &= y_3e^{-y_4(t-y_5)^2} \\
h_3(\mathbf{y}_3;t) &= y_6e^{-y_7(t-y_8)^2} \\
h_4(\mathbf{y}_4;t) &= y_9e^{-y_{10}(t-y_{11})^2}
\end{aligned}
$$

where

$$\mathbf{y}_1 = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} y_3 \\ y_4 \\ y_5 \end{bmatrix}, \quad \mathbf{y}_3 = \begin{bmatrix} y_6 \\ y_7 \\ y_8 \end{bmatrix}, \quad \mathbf{y}_4 = \begin{bmatrix} y_9 \\ y_{10} \\ y_{11} \end{bmatrix},$$

so

$$\psi(\mathbf{y}; t) = \sum_{i=1}^{4} h_i(\mathbf{y}_i; t)$$

and

$$\nabla^2_{\mathbf{y}} \psi(\mathbf{y}; t) = \begin{bmatrix} \nabla^2_{\mathbf{y}_1} h_1(\mathbf{y}_1; t) & 0 & 0 & 0 \\ 0 & \nabla^2_{\mathbf{y}_2} h_2(\mathbf{y}_2; t) & 0 & 0 \\ 0 & 0 & \nabla^2_{\mathbf{y}_3} h_3(\mathbf{y}_3; t) & 0 \\ 0 & 0 & 0 & \nabla^2_{\mathbf{y}_4} h_4(\mathbf{y}_4; t) \end{bmatrix}.$$

Hessian structures like this can be utilised to develop efficient numerical methods.

A `Maple` text file to define $\phi(x; t)$, $\psi(\mathbf{x}; t)$ and find the gradient and Hessian is in the file `stex1.txt`.

1.3.5 Exercises

1. For each of the problems 1 to 4 in Exercises 1.1.4.
   (a) Say as much as you can about the structure of the problem.
   (b) Do global extrema exist for the problem?
   (c) Use the optimization packages available in the class account to solve the problem. Interpret the solution for a manager who is NOT a mathematician.
2. Figure 1.3.3 illustrates a network with nodes $\{1, \ldots, 6\}$ and arcs $\{1, \ldots, 9\}$.

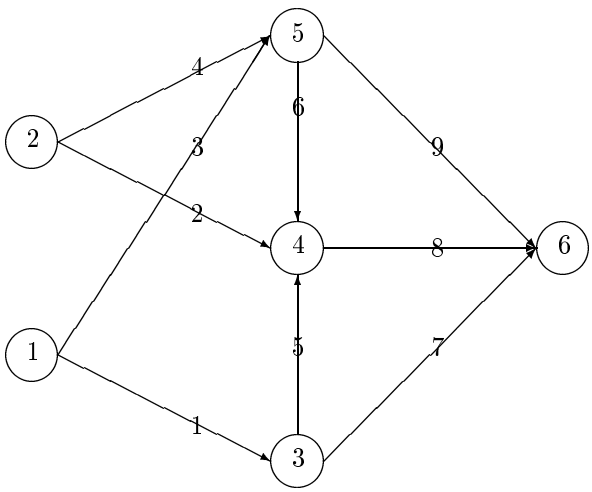


Figure 1.3.3: Network with 6 nodes and 9 arcs

   (a) What is the node-arc incidence matrix for this network?
   (b) If there is an external inflow of 10 units at node 1 and 20 units at node 2, and an outflow of 30 units at node 6, what is the system of equations representing conservation of flow?

1.4 Practical Issues

This section discusses some important issues that arise when solving practical problems.

1.4.1 Rounding errors

Many problems are finally solved by a computer program implementing a direct or iterative method. This involves calculations with floating point numbers, which are only approximations to real numbers. The possible effects of the rounding errors which are made whenever a real number is stored as a nearby floating point number must be considered. Most workstations and PCs use IEEE arithmetic in which the floating point number systems have the characteristics outlined in Table 1.4.1. The relative machine

Number system	ϵ	*tiny*	*huge*	*nsf*
IEEE 754 (single precision)	5.96e-08	1.18e-38	3.40e+38	7
IEEE 754 (double precision)	1.11e-16	2.23e-308	1.80e+308	15

Table 1.4.1: Derived parameters of some floating point systems

precision ϵ is the smallest positive number such that $1+\epsilon > 1$. The smallest non-zero number that can be represented in the floating point number system is *tiny*, while the largest number that can be represented is *huge*. The number of significant decimal digits in a floating point number is *nsf*. If a calculation produces a quantity larger than the *huge* then an *overflow* occurs, which is a fatal error. If a calculation produces a quantity smaller than *tiny* then an *underflow* to zero typically occurs. More information on floating point arithmetic can be found in [Gol91] and many texts on numerical analysis.

Example 1.4.1 (Errors from limited precision arithmetic)
This example comes from Hansen [Han92] who uses it to illustrate the use of interval arithmetic. Consider the function of two variables

$$f(\mathbf{x}) = \frac{1335}{4}x_2^6 + x_1^2(11x_1^2x_2^2 - x_2^6 - 121x_2^4 - 2) + \frac{11}{2}x_2^8 + \frac{x_1}{2x_2}. \tag{1.4.1}$$

Let $\bar{\mathbf{x}} = [\ 77617 \quad 33096\ ]^T$. *Using exact arithmetic (for example* `Maple` *or* `Mathematica`*) then*

$$f(\bar{\mathbf{x}}) = -\frac{54767}{66192} \approx -0.82739605994682136814.$$

However if floating point arithmetic is used to evaluate $f(\hat{\mathbf{x}})$ *the answer will be wrong. For example specifying the fractional constants in (1.4.1) as decimals, so*

$$f(\mathbf{x}) = 333.75x_2^6 + x_1^2(11x_1^2x_2^2 - x_2^6 - 121x_2^4 - 2) + 5.5x_2^8 + \frac{x_1}{2x_2},$$

then `Maple` *gives* $f(\hat{\mathbf{x}}) = 10^{27}$. *This is because the default working precision in* `Maple` *is 10 decimal digits. Other incorrect answers are given by programs in Fortran or C (using either single or double precision IEEE/754 arithmetic the calculated value of* $f(\hat{\mathbf{x}})$ *is approximately* -1.18059×10^{21}*), by spreadsheets such as Excel and Lotus-123, and by matrix packages such as* MATLAB. *The problem is the limited precision used in floating point number systems cannot store enough information to produce a correct answer when similar large numbers are subtracted (*catastrophic cancellation*).*

It is essential to know the limits of the data types, the computers and computer arithmetic that are being used. The numerical stability, that is how the calculations affect the rounding error, of the linear algebra that forms the basis of an algorithm is also important. Higham [Hig96] gives many interesting examples of the effects of rounding errors.

1.4.2 Testing for zero

There are many situations when one needs to test if a scalar, vector or matrix is zero, or test if two scalars, vectors or matrices are equal. If all arithmetic is exact this can be done by checking if the norm of the difference is zero, for example

$$\alpha = \bar{\alpha} \iff |\alpha - \bar{\alpha}| = 0, \tag{1.4.2a}$$

$$\mathbf{x} = \overline{\mathbf{x}} \iff \|\mathbf{x} - \overline{\mathbf{x}}\| = 0, \tag{1.4.2b}$$

$$G = \bar{G} \iff \|G - \bar{G}\| = 0. \tag{1.4.2c}$$

However if the scalars, vectors or matrices contain real numbers represented in the floating point number system of a computer, then the limited precision of floating point arithmetic must be taken into account. Let $\tau(n, \epsilon)$ represent a tolerance which depends on the relative machine precision ϵ and the size n of the vectors or matrices. Then in floating point arithmetic (1.4.2) should be replaced by

$$\alpha \approx \bar{\alpha} \iff |\alpha - \bar{\alpha}| \leq \tau(\epsilon), \tag{1.4.3a}$$
$$\mathbf{x} \approx \overline{\mathbf{x}} \iff \|\mathbf{x} - \overline{\mathbf{x}}\| \leq \tau(n, \epsilon), \tag{1.4.3b}$$
$$G \approx \bar{G} \iff \|G - \bar{G}\| \leq \tau(n, \epsilon). \tag{1.4.3c}$$

The tests (1.4.3) do not take any account of the magnitude of the quantities α, $\mathbf{x}$ or G, and the fact that ϵ is a *relative* machine precision. Instead of using the absolute errors it is better to use the relative errors, especially when the magnitude of the quantities being test for equality is large. This leads to the tests

$$\alpha \approx \bar{\alpha} \iff |\alpha - \bar{\alpha}| \leq \tau(\epsilon) \max\{1, |\alpha|\}, \tag{1.4.4a}$$
$$\mathbf{x} \approx \overline{\mathbf{x}} \iff \|\mathbf{x} - \overline{\mathbf{x}}\| \leq \tau(n, \epsilon) \max\{1, \|\mathbf{x}\|\}, \tag{1.4.4b}$$
$$G \approx \bar{G} \iff \|G - \bar{G}\| \leq \tau(n, \epsilon) \max\{1, \|G\|\}. \tag{1.4.4c}$$

Any vector or matrix norm can be used, although the choice of norm can affect the choice of $\tau(n, \mathbf{x})$. For example if the 2-norm $\|\mathbf{x}\| = [\sum_{i=1}^n x_i^2]^{\frac{1}{2}}$ or 1-norm $\|\mathbf{x}\| = \sum_{i=1}^n |x_i|$ are used then rounding error can accumulate when the sum is calculated an $\tau(n, x)$ should take into account the number n of elements in the vector $\mathbf{x}$. However when using the ∞-norm $\|\mathbf{x}\| = \max_{i=1,\ldots,n} |x_i|$ the number of elements does not directly affect what is a suitable choice for τ.

Another issue, especially with matrix norms, is the cost of calculating the norm. This means that for matrices the 1-norm and ∞-norm are more commonly used than the 2-norm (which required the calculation of the largest magnitude eigenvalue).

1.4.3 Differentiability

The mathematical model may involve functions that do not satisfy the assumptions for classical analysis. Classically the functions defining the mathematical model are assumed to be at least once continuously differentiable. $C^k(\mathbb{R}^n)$ is the set of all functions with k continuous derivatives on $\mathbb{R}^n$. Many numerical optimization methods are based on quadratic models, which can be justified in a neighbourhood of a solution if $f \in C^2(\mathbb{R}^n)$.

However some important applications give rise to nonsmooth functions, where $f \in C^0(\mathbb{R}^n)$ but $f \notin C^1(\mathbb{R}^n)$.

Example 1.4.2 (Call option)

Suppose you have an call option to buy a BHP share. If you choose to exercise the option at the expiry date, then by paying a strike price \$K you will be able to purchase a BHP share worth \$S. On the other hand you can choose not to exercise the option, and throw it away. The value of the option is then

$$V(S) = \max\{\ 0, S - K\ \}.$$

$V : \mathbb{R} \to \mathbb{R}$ is a piecewise linear function of the share price S, with a derivative discontinuity at $S = K$. Thus classical methods which assume the function is at least continuously differentiable are unlikely to be applicable in this case.

Nonsmooth optimization problems, where the function is continuous but not necessarily continuously differentiable, typically arise when there is an inner optimization. Examples include the 1-norm

$$f(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|_1 = \sum_{i=1}^m |r_i(\mathbf{x})| = \sum_{i=1}^m \max[\ r_i(\mathbf{x}), -r_i(\mathbf{x})\ ]$$

which is potentially nonsmooth at any point $\mathbf{x}$ where at least one $r_i(\mathbf{x}) = 0$, and the max function

$$f(\mathbf{x}) = \max_{i=1,\ldots,m} r_i(\mathbf{x})$$

which is potentially nonsmooth at any point where more than one function $r_i(\mathbf{x})$ achieves the maximum. Other examples include the extreme eigenvalues of matrix valued functions and decomposition of large scale problems.

It is also possible to get functions which are once continuously differentiable, but not twice continuously differentiable. An example of this is the semi-variance

$$f(\mathbf{x}) = \sum_{i=1}^{m} p_i [\, \max\{0, r_i(\mathbf{x})\} \,]^2$$

which arises as a one-sided measure of risk in portfolio optimization problems.

1.4.4 Calculating derivatives

Even if the problem functions are known to be twice continuously differentiable, examples arise where the calculation of $f(\mathbf{x})$ is very expensive, the evaluation of $\nabla f(\mathbf{x})$ is very difficult and very expensive, and the evaluation of $\nabla^2 f(\mathbf{x})$ is nearly impossible. For example in optimal control theory an objective function could be

$$f(\mathbf{x}) = \int_0^T p(x, y, t) dt \tag{1.4.5}$$

where the integral has to be evaluated numerically. Moreover y may specified by a (system) of differential equations

$$\frac{dy}{dt} = q(\mathbf{x}, y, t) \quad \text{for} \quad t \in [0, T] \tag{1.4.6}$$

where both the functions p and q depend nonlinearly on $\mathbf{x}$ and $\mathbf{y}$. Evaluating $f(\mathbf{x})$ and $\nabla f(x)$ requires a lot of work, while $\nabla^2 f(\mathbf{x})$ can be just too hard to obtain.

A symbolic manipulation (computer algebra) package, such as `Maple`, `Mathematica`, `Axiom`, `Macsyma`, `Reduce` or `Derive` can be used to obtain expressions for the gradient and Hessian. These packages can frequently produce Fortran code as output.

Another possibility is to use *finite differences* to numerically approximate the derivatives required for the gradient and Hessian. This is discussed in more detail in Section 1.4.5.

There is also a growing use of *Automatic differentiation* [GC91], where you write a function (for example in Fortran or C) to specify $f(\mathbf{x})$ and a program then writes a function to calculate the derivatives of $f(\mathbf{x})$. The forward difference approximation (1.4.7) seems to indicate that an evaluation of the gradient requires n extra function evaluations. The key to automatic differentiation is that it can be done with a cost bounded by $Kcost(f(\mathbf{x}))$, where K is a small constant (≈ 3) which is independent of the number of variables n. Bischoff and Carle received the 1995 J. H. Wilkinson Numerical Software Prize for the development of ADIFOR 2.0 (ADIFOR stands for Automatic Differentiation of Fortran).

Expressions for the gradient $\nabla f(\mathbf{x})$ and the Hessian $\nabla^2 f(x)$ should always be checked, as a common source of error is to have inconsistent expressions for $f(\mathbf{x})$, $\nabla f(x)$, and $\nabla^2 f(\mathbf{x})$. The simplicity of the problems seen in class is frequently misleading.

1.4.5 Finite difference approximations

Finite difference approximations can be used to estimate derivatives. A forward difference approximation of the first derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ is

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} + O(h) \tag{1.4.7}$$

requires an additional function evaluation for each component of the gradient. A central difference approximation to the first derivative

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h} + O(h^2) \tag{1.4.8}$$

has a smaller truncation error of $O(h^2)$, but requires two additional function evaluations for each component of the gradient. A central difference approximation of the second derivative is

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i} = \frac{f(\mathbf{x}+h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x}-h\mathbf{e}_i)}{h^2} + O(h^2). \tag{1.4.9}$$

These finite difference approximations involve the subtraction of two nearly equal numbers (often called *catastrophic cancellation*) which can amplify the effects of rounding error. The appropriate choice of h to balance rounding error and truncation error is an important issue in the numerical approximation of derivatives, which is an intrinsically ill-conditioned problem. Table 1.4.2 illustrates the use of difference approximations to estimate $f'(x)$ and $f''(x)$ for $f(x) = \log(x)$ at $x = 2$. A MATLAB version of this example is in the file `fdiff.m`. Note that when h is sufficiently small the differences $f(\mathbf{x}+h\mathbf{e}_i) - f(\mathbf{x})$

	Forward Difference approximation to $f'(x)$		Central Difference approximation to $f'(x)$		Central Difference approximation to $f''(x)$	
h	Approximation	Error	Approximation	Error	Approximation	Error
1.0e+00	0.40546510811	9.4e-02	0.54930614433	-4.9e-02	-0.28768207245	3.8e-02
1.0e-01	0.48790164169	1.2e-02	0.50041729278	-4.2e-04	-0.25031302181	3.1e-04
1.0e-02	0.49875415110	1.2e-03	0.50000416673	-4.2e-06	-0.25000312505	3.1e-06
1.0e-03	0.49987504165	1.2e-04	0.50000004167	-4.2e-08	-0.25000003112	3.1e-08
1.0e-04	0.49998750042	1.2e-05	0.50000000042	-4.2e-10	-0.24999998738	-1.3e-08
1.0e-05	0.49999875001	1.2e-06	0.50000000001	-8.8e-12	-0.24999891046	-1.1e-06
1.0e-06	0.49999987506	1.2e-07	0.50000000001	-1.4e-11	-0.24991120284	-8.9e-05
1.0e-07	0.49999998697	1.3e-08	0.49999999974	2.6e-10	-0.25535129566	5.4e-03
1.0e-08	0.49999999696	3.0e-09	0.49999999696	3.0e-09	0.00000000000	-2.5e-01
1.0e-09	0.50000004137	-4.1e-08	0.50000004137	-4.1e-08	0.00000000000	-2.5e-01
1.0e-10	0.50000004137	-4.1e-08	0.50000004137	-4.1e-08	0.00000000000	-2.5e-01
1.0e-11	0.50000004137	-4.1e-08	0.50000004137	-4.1e-08	0.00000000000	-2.5e-01
1.0e-12	0.50004445029	-4.4e-05	0.50004445029	-4.4e-05	0.00000000000	-2.5e-01
1.0e-13	0.49960036108	3.9e-04	0.49960036108	4.0e-04	0.00000000000	-2.5e-01
1.0e-14	0.51070259133	-1.0e-02	0.50515147620	-5.2e-03	0.00000000000	-2.5e-01
1.0e-15	0.44408920985	5.5e-02	0.49960036108	4.0e-04	0.00000000000	-2.5e-01
1.0e-16	0.00000000000	5.0e-01	0.00000000000	5.0e-01	0.00000000000	-2.5e-01
1.0e-17	0.00000000000	5.0e-01	0.00000000000	5.0e-01	0.00000000000	-2.5e-01

Table 1.4.2: Difference approximations using double precision arithmetic

and $f(\mathbf{x}+h\mathbf{e}_i) - f(\mathbf{x}-h\mathbf{e}_i)$ are numerically zero. For the forward difference approximation (1.4.7) the best accuracy is obtained by minimizing the sum of the $O(h)$ truncation error plus the $O(\frac{\epsilon}{h})$ rounding error. This gives an optimal $h \approx \epsilon^{\frac{1}{2}} \approx 10^{-8}$ in double precision arithmetic. Here ϵ is the relative machine precision (see Table 1.4.1). For the central difference approximation (1.4.8) the error is minimized for $h \approx \epsilon^{\frac{1}{3}} \approx 10^{-5}$. This is clearly illustrated in Figure 1.4.1

For large problems, where the Hessian or Jacobian is typically sparse, the sparsity can be exploited to reduce the amount of work required to obtain a difference approximation (see Powell and Reid for an early example).

1.4.6 Computational complexity

The numerical problems are often very large, so the efficiency of the solution procedure (both in terms of the amount of time and space required) is very important. It is essential that full use is made of the structure of the problem, both in the analysis of the problem and in the use of numerical methods. Key limiting factors, which are functions of the size of the problem, are:

1. The data required to implement a solution method.

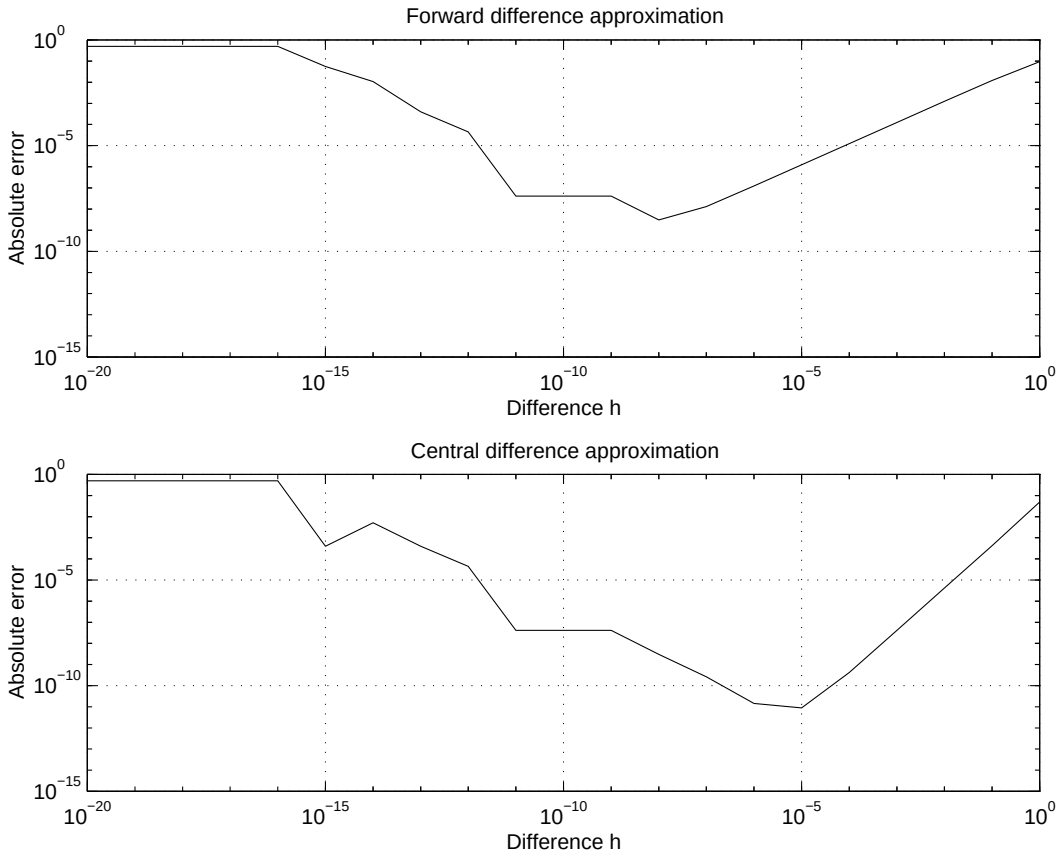


Figure 1.4.1: Errors in finite difference approximations to $f'(x)$ at $x = 2$

2. The time required to solve the problem.

It is essential to have a good idea how these quantities grow as the size of the problem increases. Let n be a measure of the problem size and let $\phi(n)$ be a quantity which depends on the problem size (for example the amount of storage required or the time to perform a given calculation).

Definition 1.4.3 (Order notation)

$$\phi(n) = O(n^\alpha) \iff \frac{\phi(n)}{n^\alpha} \to K \text{ as } n \to \infty \text{ where } 0 < K < \infty, \tag{1.4.10}$$

$$\phi(n) = o(n^\alpha) \iff \frac{\phi(n)}{n^\alpha} \to 0 \text{ as } n \to \infty \tag{1.4.11}$$

Example 1.4.4 (Storing a symmetric matrix) *When storing an n by n real symmetric matrix A, only the upper (or lower) triangle must be stored, with the other elements defined by symmetry $A_{ij} = A_{ji}$. Storing the upper (or lower) triangle requires $n(n+1)/2$ numbers. Typically the stored triangle is packed into a vector $u \in \mathbb{R}^{n(n+1)/2}$ with*

$$u_{(i-1)n+j} = a_{ij} \quad j = 1, \ldots, n, \quad i = 1, \ldots, j.$$

Using double precision arithmetic each number requires 8 bytes of memory. Thus storing an n by n symmetric matrix requires

$$8 \times \frac{n(n+1)}{2} = 4n^2 + O(n) \quad \text{bytes}$$

of storage. If a computer has a memory of M megabytes then the largest matrix that can be stored (ignoring essential overheads like the operating system) is

$$n \approx 500\sqrt{M}.$$

Thus in a well equipped PC (or a poorly equipped workstation) with 16 Megabytes of memory the largest possible value of n is around 2000.

Alternatively the amount of memory required to store a 100,000 by 100,000 real symmetric matrix is approximately

$$4n^2 = 4 \times 10^{10} \text{ bytes} \approx 4 \times 10^4 \text{ Mbytes} \approx 40 \text{ Gbytes}.$$

This is currently around the capacity of the largest supercomputers, which have memory of several GBytes. (A 32 bit operating system can directly address a maximum of 4 GBytes, but 64 bit operating systems can address much more).

Typically large matrices are *sparse*; that is they have a large number of elements which are known to be zero. For sparse matrices only the locations and values of the elements which may be non-zero need to be stored. This saves space, but complicates the data structures, and often the algorithm.

The complexity of the calculations involved in solving a problem may grow rapidly as a function of the problem size. Knowledge of the complexity of the calculations involved means that an estimate of the time required to solve the problem can be made.

Example 1.4.5 (Complexity of matrix multiplication) *Calculating the matrix-matrix product $C = AB$, where A, B, and C are all n by n matrices, by*

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad \text{for } i, j = 1, \ldots, n \tag{1.4.12}$$

takes n^3 multiplications and n^3 additions. Thus there is a an algorithm for calculating the matrix-matrix product of two n by n matrices which is a (cubic) polynomial of a measure n of the problem size.

In fact there are asymptotically faster algorithms with complexity $O(n^\alpha)$ where $2 < \alpha < 3$. For example Strassen's algorithm [Str69] takes $O(n^{log_2(7)}) \approx O(n^{2.81})$ flops. Several other more complicated algorithms have been proposed [Pan84a, Pan84b, LPS92, BP94]. The current record is an $O(n^{2.376})$ algorithm proposed by Winograd and Coppersmith [CW87] in 1987. However the constant multiplying the $n^{2.376}$ means this is not a practical algorithm. It is an open question if there exists an $O(n^{2+\eta})$ method for any smaller $\eta > 0$. A good general reference on complexity theory is the book by Traub, Wasilkowski and Wozniakowski [TWW88].

Another key issue is the numerical stability of these algorithms (Strassen's algorithm is numerically stable [Hig96, DH92]). All but the simplest of these algorithms is too complicated to implement efficiently [Bai88, BLS91, LPS92].

On the other hand consider the *Travelling Salesman Problem* (TSP) [LLKS85], where a salesman must visit each of n cities exactly once and then return to the city where they started. There is a cost of c_{ij} in travelling from city i to city j. The problem is to minimize the total cost of visiting all cities. Simply enumerating all possible routes and calculating the cost of each route involves $n!$ operations. With any physically realistic computer this is impossible for quite moderate values of n. Here the complexity of solving the travelling salesman problem by simple enumeration is $n!$, which grows faster than any polynomial of the problem size n.

Example 1.4.6 (Time for matrix multiplication) *The fastest single processor has a clock period slightly more than $1ns$ ($1ns = 10^{-9}$ secs). This corresponds to a processor rated at 1000 MHz. Assume one floating point addition and one floating point multiplication can be done each clock period. How long will it take to calculate $C = AB$ where A, B, C are n by n matrices with $n = 100,000$?*

From (1.4.12) calculating C takes n^3 multiplications and n^3 additions. This will take $10^{-9}n^3$ seconds, on a computer with a $1ns$ clock period. For $n = 10^5$ the time is

$$T = 10^{-9} \times 10^{15} = 10^6 \text{ seconds} \approx 11.5 \text{ days},$$

This is clearly unrealistic. In practice matrices of size $n = 100,000$ are very sparse, with a low ($< 1\%$) sparsity. The sparse structure of the matrices can be used to dramatically decrease the time taken to calculate C.

Although the calculation time $T = O(n^3)$ in Example 1.4.6 is a polynomial in the problem size n, the cubic grows rapidly for large n. The computational complexity of common matrix operations is listed in Table A.9.1 on page 247.

For some problems, known as NP-complete problems, there is no known polynomial time algorithm for finding a solution. Moreover NP-complete problems are equivalent in the sense that if a polynomial time algorithm can be found for one NP-complete problem then polynomial time algorithms can be found for all NP-complete problems (see Gary and Johnson [GJ79] for more information). This means that it is totally impractical, even on the fastest supercomputers, to try to find the exact solution of NP-hard problems for even moderately sized problems. Instead heuristics must be used to find an approximate solution. Further discussion on combinatorial optimization can be found in [LLKS85, NW88, PS82, CCPS98].

1.4.7 Efficient implementations

Many algorithms depend critically on efficient stable numerical linear algebra routines. Fortunately most of the building blocks are available in the Basic Linear Algebra Subroutines (BLAS) [LHKK79]. The suitability of matrix operations for vectorization, particularly in an interpreted languages like MATLAB or on advanced RISC or vector architectures, and parallelization are often key issues.

For large matrices the way in which vector and matrix operations are implemented can make a considerable difference. An example of different implementations of the matrix-matrix product $C = AB$ is given in Example A.9.1 on page 247 of Appendix A. Fortunately the level of detail considered in Example A.9.1 is rarely necessary as mathematical libraries with good implementations of the BLAS are available from the computer manufacturer or as part of general purpose mathematical libraries such as NAG [NAG] and IMSL [Vis]. The vendors of most workstations also usually provide optimized versions of the BLAS for their particular architectures. The three main levels of the BLAS are outlined in Table A.9.3 on page 249 of Section A.9. Thus using the BLAS for matrix operations provides an efficient portable way of implementing algorithms that are based on matrix operations.

Efficient routines for a wide range of matrix operations are available through the LAPACK project [ABB+95] which uses the level III BLAS. LAPACK is a successor to the widely used LINPACK [DBMS79] and EISPACK [SBD+67, GBDM77] libraries. Matrix languages such as MATLAB also include efficient versions of many essential matrix operations.

1.4.8 Desired solution

Different solution characteristics may be important depending on the physical situation which produced the mathematical problem. Two extreme cases are:

1. A solution $\mathbf{x}^*$ of the mathematical model must be found as accurately as possible. This requires that the data and functions specifying the problem can be evaluated accurately.

2. Any point within 10% of the solution is satisfactory. There may even be uncertainties in the evaluation of the model functions – for example from experimental error, from simulation of a complex system or from an inner numerical model which can only be solved approximately.

The desired accuracy affects both the convergence criteria used and the iterative method used. If accurate solutions are required then a method with fast local convergence is appropriate.

1.4.9 Exercises

1. Consider the function $f(x) = e^{x^2}$ where $x \in \mathbb{R}$.

   (a) What is the largest value of x that can be used to evaluate f without causing an overflow in single precision IEEE arithmetic?

(b) What is the largest value of x that can be used to evaluate f without causing an overflow in double precision IEEE arithmetic?

(c) For $f(x) = e^{-x^2}$ what is the largest value of x that will produce a non-zero value (assuming underflow to zero) in single and double precision IEEE arithmetic.

2. The fastest single processor supercomputer has a clock speed of around $1ns$ ($1ns = 10^{-9}$ seconds). Assume one addition and one multiplication can be done every clock cycle.

   (a) What are the largest matrices $A, B \in \mathbb{R}^{n \times n}$ that can be multiplied in

      i. 1 second.

      ii. 1 minute.

      iii. 1 day.

   (b) What is the largest Travelling Salesman Problem that can be solve by simple enumeration in

      i. 1 second.

      ii. 1 minute.

      iii. 1 day.

   Comment on the differences between your answers to parts (a) and (b).

1.5 Algorithms

Numerical algorithms require a starting point (initial guess) $\mathbf{x}^{(1)}$. They then typically try to generate a sequence of points $\{\mathbf{x}^{(k)}\}$ which progresses steadily towards a neighbourhood of a local minimizer $\mathbf{x}^*$, and then converges rapidly to the point $\mathbf{x}^*$ itself.

Many methods are *line search methods*. At a point $\mathbf{x}^{(k)}$ a search direction $\mathbf{d}^{(k)}$ is generated. The next iterate $\mathbf{x}^{(k+1)}$ is then chosen by performing an (approximate) line search (one-dimensional minimization)

$$\alpha^{(k)} = \text{argmin } f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) \quad \alpha \geq 0. \tag{1.5.1}$$

The next iterate is then

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)}.$$

As the standard problem is a minimization problem we want *descent methods* where

$$f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}).$$

In a line search method this can be guaranteed if $\mathbf{d}^{(k)}$ is a *descent direction*, that is

$$\mathbf{d}^{(k)^T} \mathbf{g}^{(k)} < 0.$$

1.5.1 Global convergence

A method is *globally convergent* to $\mathbf{x}^*$ if given ANY starting point $\mathbf{x}^{(1)}$ the method produces a sequence of points $\{\mathbf{x}^{(k)}\}$ such that $\mathbf{x}^{(k)} \to \mathbf{x}^*$ as $k \to \infty$. Global convergence does NOT imply that $\mathbf{x}^*$ is a global minimizer. Except in certain special cases, for instance when the feasible region is convex and f is a convex function, we have to be satisfied with finding a local minimizer. The problem of finding a global minimizer, usually referred to as global optimization, is another area of active research.

1.5.2 Local convergence

Let $\{\mathbf{x}^{(k)}\}$ be a sequence which converges to $\mathbf{x}^*$ as $k \to \infty$, and let

$$\lim_{k \to \infty} \frac{||\mathbf{x}^{(k+1)} - \mathbf{x}^*||}{||\mathbf{x}^{(k)} - \mathbf{x}^*||^\nu} = \beta. \tag{1.5.2}$$

The *rate of convergence* is

1. *Linear* (or first order) with rate β if $\nu = 1$ and $0 < \beta < 1$.
2. *Superlinear* if $\nu = 1$ and $\beta = 0$ or $\nu > 1$ and $\beta < \infty$.
3. *Quadratic* (or second order) if $\nu = 2$ and $\beta < \infty$.

The *order of convergence* is the largest number ν such that which produces a finite limit β. Thus a sequence $\{\mathbf{x}^{(k)}\}$ of points in $\mathbb{R}^n$ converges to $\mathbf{x}^*$

- linearly if there exists a constant $\beta < 1$ and integer $\bar{k}$ such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \beta \|\mathbf{x}^{(k)} - \mathbf{x}^*\| \quad \forall\, k \geq \bar{k}.$$

- superlinearly if there exist constants $\nu > 1$, $\beta < \infty$, and an integer $\bar{k}$, such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \beta \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{\nu} \quad \forall\, k \geq \bar{k}.$$

- quadratically if there exists a constant $\beta < \infty$, and an integer $\bar{k}$, such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \beta \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{2} \quad \forall\, k \geq \bar{k}.$$

ν	1.0	1.0	1.6	2.0
β	0.99	0.2	1.0	1.0
k	$\|\mathbf{e}^{(k)}\|$	$\|\mathbf{e}^{(k)}\|$	$\|\mathbf{e}^{(k)}\|$	$\|\mathbf{e}^{(k)}\|$
1	5.00e-001	5.00e-001	5.00e-001	5.00e-001
2	4.95e-001	1.00e-001	3.30e-001	2.50e-001
3	4.90e-001	2.00e-002	1.70e-001	6.25e-002
4	4.85e-001	4.00e-003	5.85e-002	3.91e-003
5	4.80e-001	8.00e-004	1.06e-002	1.53e-005
6	4.75e-001	1.60e-004	6.97e-004	2.33e-010
7	4.71e-001	3.20e-005	8.90e-006	5.42e-020
8	4.66e-001	6.40e-006	8.30e-009	2.94e-039
9	4.61e-001	1.28e-006	1.18e-013	8.64e-078
10	4.57e-001	2.56e-007	2.06e-021	7.46e-155
11	4.52e-001	5.12e-008	7.97e-034	5.56e-309
12	4.48e-001	1.02e-008	1.10e-053	0.00e+000

Table 1.5.1: Convergence of various orders and rates

If the convergence is linear then the rate β is critical. Let

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^* \quad \text{and} \quad \epsilon^{(k)} = \|\mathbf{e}^{(k)}\|$$

be the error in the kth iterate $\mathbf{x}^{(k)}$ and the norm of the error respectively. Then, for k sufficiently large, a linearly convergent method has errors which satisfy

$$\epsilon^{(k+1)} \approx \beta \epsilon^{(k)}. \tag{1.5.3}$$

Thus the error is being reduced by a factor $0 < \beta < 1$ on each iteration. If β is close to zero the convergence appears rapid, while if β is close to 1 then the convergence is very slow. A superlinearly convergent sequence has a faster rate of convergence than any linearly convergent sequence. The definition of superlinear convergence includes all the cases where $\nu > 1$ and $\beta < \infty$. Table 1.5.1 illustrates the norms of the errors $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$ for linear convergence with $\beta = 0.99$, $\beta = 0.2$, superlinear convergence with $\nu = 1.6$ and quadratic convergence.

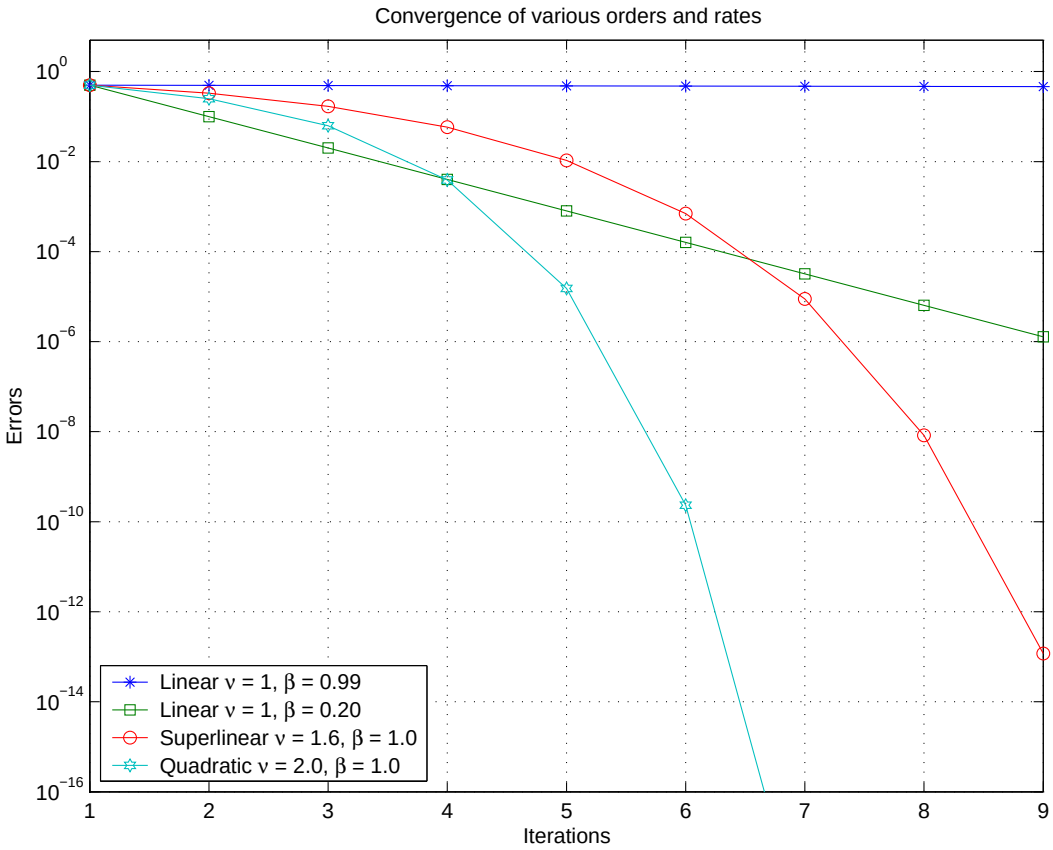


Figure 1.5.1: Convergence of various orders and rates

A linearly convergent sequence satisfies (1.5.3), so $\epsilon^{(k+1)} \approx \beta^k \epsilon^{(1)}$, and

$$\log\left(\epsilon^{(k+1)}\right) \approx k\log(\beta) + \log\left(\epsilon^{(1)}\right). \tag{1.5.4}$$

Thus a plot of iterations k versus the logarithms of the norms $\log(\epsilon^{(k)})$ of the errors should be a straight line, with slope $\log(\beta)$. If the solution $\mathbf{x}^*$ is known calculating and plotting the errors is a good way to verify the rate of convergence of a method. The different rates of convergence from Table 1.5.1 are plotted in Figure 1.5.1. These errors are plotted in Figure 1.5.1.

It is important to note that the rate of convergence is an *asymptotic* property of a method, and it only indicates how quickly $\mathbf{x}^{(k)}$ converges to $\mathbf{x}^*$ ONCE $\mathbf{x}^{(k)}$ is close to $\mathbf{x}^*$. The rate of convergence gives no idea how the iterates $\mathbf{x}^{(k)}$ will behave far away from $\mathbf{x}^*$.

1.5.3 Convergence conditions

Suppose $\{\mathbf{x}^{(k)}\}$ is a sequence of points which converges to a local minimizer $\mathbf{x}^*$. Ideally we would stop the iteration when either or both of the conditions

$$||\mathbf{x}^{(k)} - \mathbf{x}^*|| \le \epsilon_x, \qquad |f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*)| \le \epsilon_f. \tag{1.5.5}$$

Here $\epsilon_x > 0$ and $\epsilon_f > 0$ are small constants reflecting the desired accuracy in the solution point and the function value. Unfortunately $\mathbf{x}^*$ and $f(\mathbf{x}^*)$ are usually unknown, so this is impossible. As a compromise some combination of the following three convergence criterion can be used.

1. Difference in successive iterates:

$$||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| \le \epsilon_x \tag{1.5.6}$$

2. Difference in successive function values:

$$|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})| \leq \epsilon_f \tag{1.5.7}$$

3. Size of the gradient in unconstrained optimization:

$$||\nabla f(\mathbf{x}^{(k)})|| \leq \epsilon_g \tag{1.5.8}$$

   or the size of the gradient of the Lagrangian in constrained optimization:

$$||\nabla \mathcal{L}_x(\mathbf{x}, \lambda)|| = ||\nabla f(\mathbf{x}^{(k)}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i^{(k)} \nabla c_i(\mathbf{x}^{(k)})|| \leq \epsilon_g. \tag{1.5.9}$$

Here ϵ_x, ϵ_f, ϵ_g are positive constants chosen to reflect the desired accuracy in the solution. In constrained optimization the final iterate should be feasible, or very close to feasible.

When choosing values for the constants the rounding error of the computer used to perform the calculations must always born in mind. For example on a computer with a precision of around 7 decimal digits it is ridiculous to try for an accuracy of 10 decimal places in $\mathbf{x}^*$ by setting $\epsilon_x = 0.5 \times 10^{-10}$.

1.5.4 Affine invariance

It is highly desirable that an algorithm is invariant under affine transformations of the variables. Let

$$\mathbf{y} = A\mathbf{x} + \mathbf{b}$$

where $A \in \mathbb{R}^n$ is nonsingular. Ideally the sequence of points $\{\mathbf{y}^{(k)}\}$ generated by the algorithm applied to the problem using the $\mathbf{y}$ variables should be equivalent to the sequence $\{\mathbf{x}^{(k)}\}$ generated by applying the algorithm applied to the problem in the $\mathbf{x}$ variable, in that

$$\mathbf{y}^{(k)} = A\mathbf{x}^{(k)} + \mathbf{b}.$$

1.5.5 Comparing methods

The following attributes are commonly used to compare numerical optimization algorithms:

1. The information (function, gradient, Hessian values) required by the method.
2. The conditions required on the objective and constraint functions for the method to work. For example must $f \in C^2$ or is $f \in C^1$ enough?
3. The global convergence properties of the method.
4. The rate of convergence properties of the method. A good method usually has at least a superlinear rate of convergence.
5. The computational complexity of the method. The storage required and the number of arithmetical operations per iteration are reported as functions of the size of the problem.
6. The numerical stability of the method. How sensitive is the method to the build up of rounding errors?
7. The effects of scaling of the variables. Is the method invariant to diagonal or affine scaling of the variables?
8. Performance on a wide range of test problems, including
   (a) Problems chosen to test specific features of the algorithm.
   (b) Randomly generated problems to get an idea of the expected behaviour of the algorithm.
   (c) Problems typical of those that you really want to solve.

The growing availability and increasing power of computers is making it practical to solve much larger and more difficult optimization problems. The use of good numerical techniques, in particular efficient stable numerical linear algebra methods which do not increase the effects of round-off errors, are essential. As well as possessing good theoretical properties optimization algorithms should be tried on a range of test problems, including real life problems, randomly generated problems, and problems chosen specifically to check both the numerical and theoretical properties of the methods.

As evaluating function values and derivatives can be expensive, the number of such evaluations required to solve a particular problem by a given algorithm is commonly used as a measure of the performance of that algorithm. If the algorithm also uses gradients and Hessians then these evaluations must also be counted. Other measures of the performance of a algorithm include the number of iterations, and the computational cost (in terms of floating point operations) of each iteration.

The lack of a uniform standard for reporting an algorithm's performance on test problems was highlighted by the controversy surrounding Karmarkar's algorithm for linear programming. Klee and Minty gave an example for which the Simplex method for linear programming took 2^n iterations on a problem with n variables. Karmarkar proved his method took only a number of iterations which is a polynomial of the size of the problem. He also claimed improvements of 20-100 times over standard linear programming codes, at first these results were not replicated by other researchers. After a tremendous amount of work *Interior Point* methods, which are a generalization of Karmarkar's original proposal, are the widely accepted as the most efficient algorithm for certain classes of large sparse linear programming problems.

1.5.6 Exercises

1. Using `Maple`

   (a) Define the function
   $$f(x) = \frac{x^4}{9} - \frac{10x^3}{9} + \frac{10x^2}{3} + \frac{50x}{9}.$$

   (b) Calculate the first derivative $f'(x)$ and second derivative $f''(x)$ of $f(x)$.

   (c) Plot the function $f(x)$ and its derivatives $f'(x), f''(x)$ over the interval $[-1, 6]$.

   (d) Using a `Maple while` loop implement the following pseudo-code for Newton's method to find a minimizer of $f(x)$. Do at most $k_{max} = 20$ iterations or stop when $|f'(x)| \leq g_{tol} = 10^{-8}$. Start from $x^{(1)} = 1.1$.

   $k_{max} := 20$
   $g_{tol} := 10^{-8}$
   $k := 1$
   $x^{(k)} := 1.1$
   `Evaluate` $f(x^{(k)}), f'(x^{(k)}), f''(x^{(k)})$
   `While` $k <= k_{max}$ `and` $|f'(x^{(k)})| > g_{tol}$
   $\quad x^{(k+1)} := x^{(k)} - f'(x^{(k)})/f''(x^{(k)})$
   $\quad k := k + 1$
   $\quad$ `Evaluate` $f(x^{(k)}), f'(x^{(k)}), f''(x^{(k)})$.
   `End While`

   (e) Given that the exact solution is $x^* = \frac{5}{2}$ modify the `Maple` program to also calculate the errors $\epsilon^{(k)} = |x^{(k)} - x^*|$ for each iteration.

   (f) Estimate the rate of convergence. By default `Maple` works to a precision of 10 decimal digits for floating point calculations. To see the full behaviour of the errors more accurate calculation may be required. This can be achieved in `Maple` using the command `Digits := 20` to force floating point calculations to be done to an accuracy of 20 decimal digits.

   (g) Change the starting point to $x^{(1)} = 1$ and see what happens.

A `Maple` text file for this problem is in the file `nalg.txt` in the class account.

2. Calculate the limits and rates of convergence of the following sequences. Also calculate the first 15 terms of each sequence, and $||\mathbf{x}^{(k+1)} - \mathbf{x}^*||/||\mathbf{x}^{(k)} - \mathbf{x}^*||^\nu$ for $\nu = 1$ and 2.

   (a) $\mathbf{x}^{(k)} = \tau^{2^k}$ where $0 < \tau < 1$. Tabulate for $\tau = .99$

   (b) $\mathbf{x}^{(k)} = \tau^{2^{-k}}$ where $\tau > 0$. Tabulate for $\tau = 2.2$

   (c) $\mathbf{x}^{(k)} = \frac{1}{k^k}$. Hint: $(1 - \frac{1}{k})^k \to e^{-1}$ as $k \to \infty$.

3. Let $\{\mathbf{x}^{(k)}\}$ be a sequence such that $\mathbf{x}^{(k)} \to \mathbf{x}^*$ as $k \to \infty$. Show that the order of convergence can be estimated by

$$\nu = \frac{\log((\mathbf{x}^{(k+1)} - \mathbf{x}^*)/(\mathbf{x}^{(k)} - \mathbf{x}^*)\ )}{\log(\ (\mathbf{x}^{(k)} - \mathbf{x}^*)/(\mathbf{x}^{(k-1)} - \mathbf{x}^*)\ )}. \tag{1.5.10}$$

   Use this formula to estimate the rate of convergence from the first 15 terms of the sequences in the previous question.

Chapter 2

Convexity

A key difficulty in many applications is to determine if the problem has many local solutions or just one global solution. The theory of convex sets and convex functions provides sufficient conditions for a local minimizer to be a global minimizer. Convexity is a fundamental property of minimization problems, leading to many powerful results.

2.1 Convex sets

Definition 2.1.1 *A set* $\Omega \in \mathbb{R}^n$ *is* convex *if and only if for every* $\mathbf{x}, \mathbf{y} \in \Omega$

$$\mathbf{x}(\theta) \equiv \theta\mathbf{x} + (1-\theta)\mathbf{y} \in \Omega \quad \forall\ \theta \in [0,1]. \tag{2.1.1}$$

For a convex set Ω the line segment joining any two points of Ω lies within Ω. A convex set Ω_1 and non-convex set Ω_2 are sketched in Figure 2.1.1.

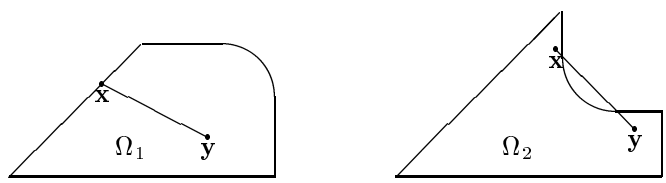


Figure 2.1.1: A convex set Ω_1 and non-convex set Ω_2

It is easy to prove that a set Ω is *not* convex simply by giving two points $\bar{\mathbf{x}}, \bar{\mathbf{y}} \in \Omega$ and a $\bar{\theta} \in [0,1]$ such that $\bar{\theta}\bar{\mathbf{x}} + (1-\bar{\theta})\bar{\mathbf{y}} \notin \Omega$. However to show that a set Ω *is* convex (2.1.1) must be established *for every* $\mathbf{x}$ and $\mathbf{y}$ in Ω.

Example 2.1.2 *Let* $\Omega = [0,1] \cup [3,5] \subset \mathbb{R}$. Ω *is not convex as* $\bar{\mathbf{x}} = 1 \in \Omega$, $\bar{\mathbf{y}} = 3 \in \Omega$ *and* $\bar{\theta} = \frac{1}{2} \in [0,1]$, *but*

$$\bar{\theta}\bar{\mathbf{x}} + (1-\bar{\theta})\bar{\mathbf{y}} = \tfrac{1}{2}1 + (1-\tfrac{1}{2})3 = 2 \notin \Omega.$$

Lemma 2.1.3 *Let* $\mathbf{a} \in \mathbb{R}^n$ *and* $\beta \in \mathbb{R}$ *be given. The half space* $H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} \le \beta\}$ *is convex.*

Proof Let $\mathbf{x}$ and $\mathbf{y}$ be arbitrary points in H and let $\theta \in [0,1]$. Then

$$\mathbf{x} \in H \iff \mathbf{a}^T\mathbf{x} \le \beta \tag{2.1.2a}$$

$$\mathbf{y} \in H \iff \mathbf{a}^T\mathbf{y} \le \beta. \tag{2.1.2b}$$

As $\theta \in [0,1]$ both $\theta \ge 0$ and $1-\theta \ge 0$. Thus multiplying (2.1.2a) by θ and (2.1.2b) by $1-\theta$ and adding gives

$$\mathbf{a}^T\mathbf{x}(\theta) = \mathbf{a}^T(\theta\mathbf{x} + (1-\theta)\mathbf{y}) = \theta\mathbf{a}^T\mathbf{x} + (1-\theta)\mathbf{a}^T\mathbf{y} \le \theta\beta + (1-\theta)\beta = \beta.$$

Hence $\mathbf{x}(\theta) \in \Omega$ for all choices of $\mathbf{x}, \mathbf{y} \in \Omega$ and $\theta \in [0,1]$, so Ω is convex. ∎

Commonly occurring examples of convex sets are

1. The set $\Omega = \{\mathbf{a}\}$ consisting of a single element $\mathbf{a} \in \mathbb{R}^n$.
2. The line $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \bar{\mathbf{x}} + \alpha\mathbf{d}, \alpha \in \mathbb{R}\}$, where $\bar{\mathbf{x}}$ and $\mathbf{d}$ are fixed vectors.
3. The hyperplane $H_0 = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} - \beta = 0\}$.
4. The half spaces $H_+ = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} - \beta \geq 0\}$ and $H_- = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} - \beta \leq 0\}$.
5. The whole space $\Omega = \mathbb{R}^n$.

Convex sets are frequently created from the intersection of convex sets.

Lemma 2.1.4 *The intersection of two convex sets is convex.*

Proof Let Ω_1 and Ω_2 be two convex sets, and let $\Omega = \Omega_1 \cap \Omega_2$. Let $\mathbf{x}, \mathbf{y} \in \Omega$ and $\theta \in [0,1]$ be arbitrary. As $\mathbf{x}, \mathbf{y} \in \Omega_1$ and Ω_1 is convex, so $\theta\mathbf{x} + (1-\theta)\mathbf{y} \in \Omega_1$. Similarly as $\mathbf{x}, \mathbf{y}$ belong to the convex set Ω_2, $\theta\mathbf{x} + (1-\theta)\mathbf{y} \in \Omega_2$. Thus $\theta\mathbf{x} + (1-\theta)\mathbf{y} \in \Omega_1 \cap \Omega_2$, so $\Omega = \Omega_1 \cap \Omega_2$ is convex. ∎

Note that the *union* of two convex sets is not in general convex, as Example 2.1.2 illustrates..

Definition 2.1.5 *Let Ω be a convex set. A point $\bar{\mathbf{x}} \in \Omega$ is an* extreme point *of Ω if $\mathbf{x}, \mathbf{y} \in \Omega$, $\theta \in [0,1]$, and $\bar{\mathbf{x}} = \theta\mathbf{x} + (1-\theta)\mathbf{y}$ implies that either $\theta = 0$ or $\theta = 1$ or $\mathbf{x} = \mathbf{y}$.*

Geometrically $\bar{\mathbf{x}}$ is an extreme point of Ω if there are no two distinct points $\mathbf{x}$ and $\mathbf{y}$ in Ω such that $\bar{\mathbf{x}}$ lies in the interior of the line segment joining $\mathbf{x}$ and $\mathbf{y}$.

Example 2.1.6 *For the convex set Ω sketched in Figure 2.1.2 the extreme points are the points A, B, C, and every point on the arc joining D and E, including the points D and E.*

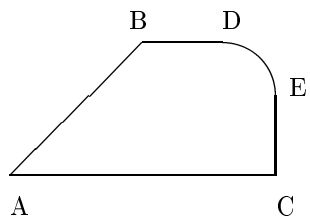


Figure 2.1.2: Extreme points

Definition 2.1.7 *The* convex combination *of $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^n$ is*

$$\mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{x}^{(i)} \quad \textit{where} \sum_{i=1}^{m} \alpha_i = 1 \textit{ and } \alpha_i \geq 0, i = 1, \ldots, m.$$

Definition 2.1.8 *The* convex hull $\operatorname{conv}\Omega$ *of a set Ω is the set of all convex combinations of points in Ω.*

Figure 2.1.3 gives an example of the convex hull of a non-convex set Ω in $\mathbb{R}^2$.

The following proposition summarises some of the properties of the convex hull of a set.

Proposition 2.1.9

1. *Any point in the convex hull of $\Omega \subseteq \mathbb{R}^n$ can be expressed as a convex combination of $m \leq n+1$ points in Ω.*

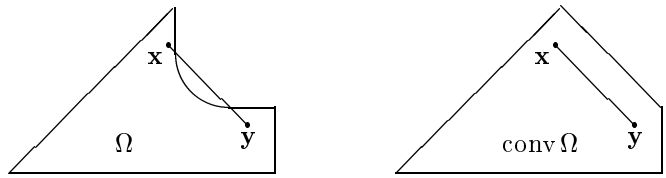


Figure 2.1.3: A non-convex set Ω and $\operatorname{conv}\Omega$

2. *The convex hull of a set is the intersection of all half-spaces containing* Ω.
3. *The convex hull of a set* Ω *is the smallest convex set containing* Ω.
4. *A convex set* Ω *is the convex hull of the extreme points of* Ω.

Definition 2.1.10 *A polyhedral convex set is a convex set which is the intersection of a* finite *number of half spaces.*

Theorem 2.1.11 (Separating Hyperplane Theorem) *Let* Ω *be a nonempty closed convex set in* $\mathbb{R}^n$ *and let* $\mathbf{z} \notin \Omega$. *Then there exists a hyperplane* $H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} = \beta\}$ *such that* $\mathbf{a}^T\mathbf{z} < \beta$ *and* $\mathbf{a}^T\mathbf{x} \geq \beta$ *for all* $\mathbf{x} \in \Omega$.

Proof Consider the problem of minimizing the continuous function $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{z}\|_2$ subject to $\mathbf{x} \in \Omega$. As

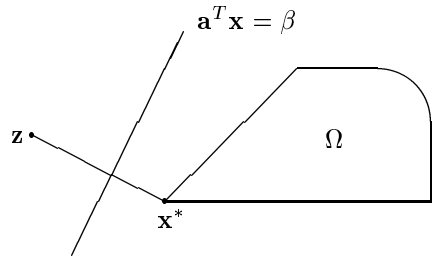


Figure 2.1.4: Separating hyperplane

Ω is non-empty there exists an $\overline{\mathbf{x}} \in \Omega$. Without loss of generality attention can be restricted to minimizing $f(\mathbf{x})$ over the closed and bounded (and hence compact) convex set

$$\overline{\Omega} = \{\, \mathbf{x} \in \Omega : \|\mathbf{x} - \mathbf{z}\|_2 \leq \|\overline{\mathbf{x}} - \mathbf{z}\|_2 \,\}.$$

As f is continuous there exists a global minimizer $\mathbf{x}^* \in \overline{\Omega} \subseteq \Omega$ to

$$\min_{\mathbf{x} \in \overline{\Omega}} ||\mathbf{x} - \mathbf{z}||_2.$$

The point $\mathbf{x}^*$ is also a global minimizer to the problem of minimizing $f(\mathbf{x})$ over the set Ω, so

$$\|\mathbf{x} - \mathbf{z}\|_2 \geq \|\mathbf{x}^* - \mathbf{z}\|_2 \quad \forall \mathbf{x} \in \Omega.$$

Let $\mathbf{x} \in \Omega$ and $\theta \in (0, 1]$ be arbitrary. As Ω is convex, the point $\theta\mathbf{x} + (1-\theta)\mathbf{x}^* \in \Omega$, so

$$||\theta\mathbf{x} + (1-\theta)\mathbf{x}^* - \mathbf{z}||_2^2 \geq ||\mathbf{x}^* - \mathbf{z}||_2^2 \qquad \forall \mathbf{x} \in \Omega, \quad \forall \theta \in [0, 1].$$

Expanding using $\|\mathbf{u}\|_2^2 = \mathbf{u}^T\mathbf{u}$ gives

$$\theta^2\mathbf{x}^T\mathbf{x} + \theta(\theta - 2)\mathbf{x}^{*T}\mathbf{x}^* + 2\theta(1-\theta)\mathbf{x}^T\mathbf{x}^* - 2\theta\mathbf{x}^T\mathbf{z} + 2\theta\mathbf{z}^T\mathbf{x}^* \geq 0.$$

Dividing by $\theta > 0$ and letting $\theta \to 0^+$ gives

$$-2\mathbf{x}^{*T}\mathbf{x}^* + 2\mathbf{x}^T\mathbf{x}^* - 2\mathbf{x}^T\mathbf{z} + 2\mathbf{z}^T\mathbf{x}^* \geq 0.$$

or

$$(\mathbf{x}^* - \mathbf{z})^T(\mathbf{x} - \mathbf{x}^*) \geq 0 \qquad \forall \mathbf{x} \in \Omega.$$

Letting $\mathbf{a} = \mathbf{x}^* - \mathbf{z}$ and $\beta = \mathbf{a}^T\mathbf{x}^*$ this implies $\mathbf{a}^T(\mathbf{x} - \mathbf{x}^*) = \mathbf{a}^T\mathbf{x} - \beta \geq 0$, or

$$\mathbf{a}^T\mathbf{x} \geq \beta \qquad \forall \mathbf{x} \in \Omega.$$

As $\mathbf{z} \notin \Omega$ and $\mathbf{x}^* \in \Omega$, $\mathbf{a}^T(\mathbf{z} - \mathbf{x}^*) = -\|\mathbf{z} - \mathbf{x}^*\|_2^2 < 0$. Hence

$$\mathbf{a}^T\mathbf{z} < \beta,$$

which completes the proof. ∎

2.1.1 Exercises

1. Establish rigourously whether the following sets are convex or not.
   (a) $\Omega = [\alpha, \beta]$ where $\alpha \leq \beta$ are any two real numbers.
   (b) $\Omega = [1, 2] \cup [3, 5]$.
   (c) $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - 2x_2 \geq 4\}$.
   (d) $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 - 2x_2 \geq 1\}$.
   (e) $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 - 2x_2 = 1\}$.
   (f) $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 - 2x_2 \leq 1\}$.
2. Prove the following sets are convex using the definition of a convex set.
   (a) $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} - \beta = 0\}$ where $\beta \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{R}^n$ are fixed.
   (b) $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T\mathbf{x} - \beta \geq 0\}$ where $\beta \in \mathbb{R}$ and $\mathbf{a} \in \mathbb{R}^n$ are fixed.
   (c) $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq \rho\}$ where $\|\mathbf{x}\|$ denotes any norm on $\mathbb{R}^n$ and $\rho \geq 0$.
   (d) $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T A\mathbf{x} \leq \rho\}$ where $\rho > 0$ and A is any n by n symmetric positive definite matrix.
3. A set $X \subseteq \mathbb{R}^n$ is *affine* if for every $\mathbf{x}, \mathbf{y} \in X$ and for every $\alpha \in \mathbb{R}$

   $$\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x}) \in X.$$

   Prove that an affine set is convex.

2.2 Convex Functions

Definition 2.2.1 *Let Ω be a convex set. and let $\mathbf{x}(\theta) = \theta\mathbf{x} + (1-\theta)\mathbf{y}$. A function f defined on Ω is*

1. convex *if and only if for every $\mathbf{x}, \mathbf{y} \in \Omega$*

$$f(\mathbf{x}(\theta)) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y}) \quad \forall \theta \in [0,1], \tag{2.2.1}$$

2. strictly convex *if and only if for every $\mathbf{x} \neq \mathbf{y} \in \Omega$*

$$f(\mathbf{x}(\theta)) < \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y}) \quad \forall \theta \in (0,1), \tag{2.2.2}$$

3. concave *if and only if for every $\mathbf{x}, \mathbf{y} \in \Omega$*

$$f(\mathbf{x}(\theta)) \geq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y}) \quad \forall \theta \in [0,1], \tag{2.2.3}$$

4. strictly concave *if and only if for every* $\mathbf{x} \neq \mathbf{y} \in \Omega$

$$f(\mathbf{x}(\theta)) > \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}) \quad \forall \theta \in (0,1), \tag{2.2.4}$$

A convex function f lies on or below the line joining the function values $f(\mathbf{x})$ and $f(\mathbf{y})$. A concave function f lies on or above the line joining the function values $f(\mathbf{x})$ and $f(\mathbf{y})$. A function f is (strictly) *concave* if and only if $-f$ is (strictly) convex. For example consider the function defined by (2.2.5).

$$f(x) = \begin{cases} -x - 0.5 & \text{if } x \leq -1; \\ (x+1)^2 + 0.5 & \text{if } -1 < x \leq 0; \\ 2.5 - e^{-x/2} & \text{if } 0 > x. \end{cases} \tag{2.2.5}$$

This function is illustrated in Figure 2.2.1. It is convex on the interval $[-2, 0]$, concave on the interval

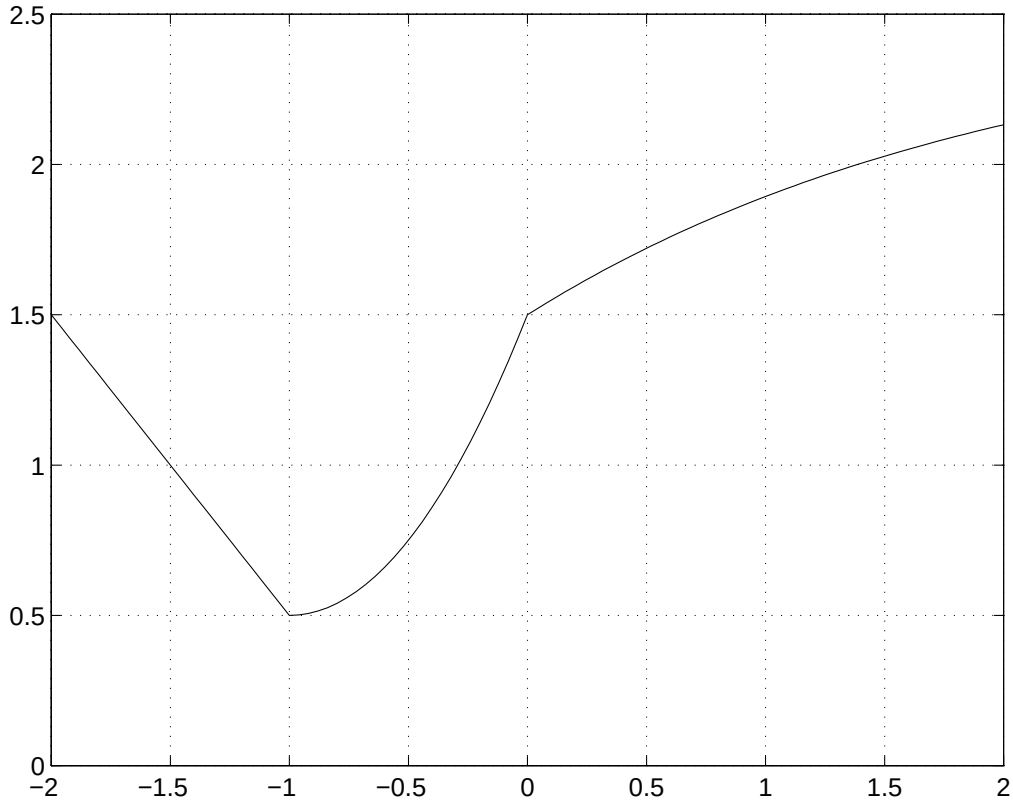


Figure 2.2.1: Convex and concave functions

$[0, 2]$, and neither convex nor concave on the interval $[-2, 2]$.

Lemma 2.2.2 *Any norm* $\|\mathbf{x}\|$ *on* $\mathbb{R}^n$ *is a convex function.*

Proof Let $\mathbf{x}$ and $\mathbf{y}$ be arbitrary points in $\mathbb{R}^n$ and let $\theta \in [0, 1]$ be arbitrary. Let $\mathbf{x}(\theta) = \theta\mathbf{x} + (1-\theta)\mathbf{y}$. Then

$$\begin{aligned} \|\mathbf{x}(\theta)\| &= \|\theta\mathbf{x} + (1-\theta)\mathbf{y}\| \\ &\leq \|\theta\mathbf{x}\| + \|(1-\theta)\mathbf{y}\| && \text{Triangle inequality} \\ &= \theta\|\mathbf{x}\| + (1-\theta)\|\mathbf{y}\| && \text{as } \theta \geq 0 \text{ and } 1-\theta \geq 0. \end{aligned}$$

Hence any norm on $\mathbb{R}^n$ is convex. ∎

Lemma 2.2.3 *Any affine function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + \beta$ is both convex and concave.*

Proof Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\theta \in [0,1]$ be arbitrary. Let $\mathbf{x}(\theta) = \theta\mathbf{x} + (1-\theta)\mathbf{y}$. Then

$$\begin{aligned} f(\mathbf{x}(\theta)) &= \mathbf{a}^T(\theta\mathbf{x} + (1-\theta)\mathbf{y}) + \beta \\ &= \theta\left(\mathbf{a}^T\mathbf{x} + \beta\right) + (1-\theta)\left(\mathbf{a}^T\mathbf{y} + \beta\right) \\ &= \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}). \end{aligned}$$

Thus $f(\mathbf{x}(\theta)) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$ so f is convex, and $f(\mathbf{x}(\theta)) \geq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$ so f is concave. ∎

Convex functions are commonly built up from other convex functions, as the following proposition illustrates.

Proposition 2.2.4 *Let $f_j : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, \ldots, m$ be convex functions. Then*

1. *the non-negative linear combination*

$$f(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j f_j(\mathbf{x}), \tag{2.2.6}$$

   where $\alpha_j \geq 0, j = 1, \ldots, m$, is convex.

2. *the maximum function*

$$f(\mathbf{x}) = \max_{j=1,\ldots,m} f_j(\mathbf{x}) \tag{2.2.7}$$

   is convex

See Exercise 2.

Definition 2.2.5 (Polyhedral convex function) *A* polyhedral convex function *is a convex function that can be expressed as the maximum of a* finite *number of affine functions, so*

$$f(\mathbf{x}) = \max_{i=1,\ldots,m} \mathbf{a}_i^T\mathbf{x} + \beta_i \tag{2.2.8}$$

The polyhedral convex function

$$f(x) = \max_{i=1,\ldots,m} f_i(x)$$

where $m = 4$ and

$$\begin{aligned} f_1(x) &= -2x - 1, \\ f_2(x) &= -x, \\ f_3(x) &= \frac{x}{2}, \\ f_4(x) &= 3x - 2. \end{aligned}$$

is sketched in Figure 2.2.2 The polyhedral convex function is not continuously differentiable at point where more than one of the component functions $f_i(x)$ achieves the maximum.

Huber's M-estimator [Hub77, Hub81, Osb85] is the convex function $h : \mathbb{R} \to \mathbb{R}$ defined by

$$h(t) = \begin{cases} \frac{1}{2\gamma}t^2 & \text{if } |t| \leq \gamma \\ |t| - \frac{\gamma}{2} & \text{if } |t| > \gamma. \end{cases} \tag{2.2.9}$$

where $\gamma > 0$ The function $h(t)$ is plotted in Figure 2.2.3 for $\gamma = 1$. This function consists of a quadratic on the interval $[-\gamma, \gamma]$ and affine pieces outside this interval. As h contains affine pieces h is not strictly

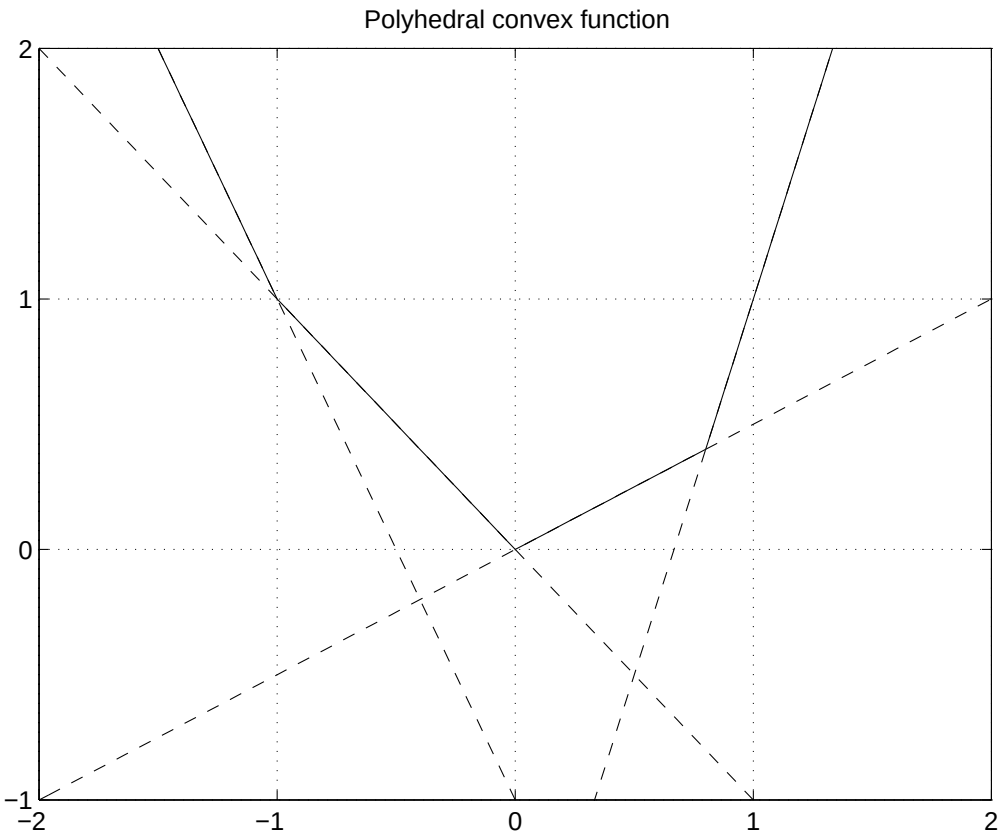


Figure 2.2.2: A polyhedral convex function

convex. The function h is used in M-estimation and robust statistics to lower the impact of large residuals. The function h is continuously differentiable, with derivative

$$h'(t) = \begin{cases} -1 & \text{if } t < -\gamma \\ \frac{t}{\gamma} & \text{if } -\gamma < t < \gamma \\ 1 & \text{if } t > \gamma, \end{cases} \tag{2.2.10}$$

but not twice continuously differentiable. The M-estimator $h(t)$ can also be thought of as a smooth (once continuously differentiable) approximation of the absolute value function $|t|$.

Definition 2.2.6 *Let $f : \mathbb{R}^n \to \mathbb{R}$. The* epigraph *of f is*

$$\text{epi } f = \left\{ \begin{bmatrix} \mathbf{x} \\ \alpha \end{bmatrix} \in \mathbb{R}^{n+1} : \alpha \geq f(\mathbf{x}) \right\} \tag{2.2.11}$$

Proposition 2.2.7 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if* epi f *is a convex set.*

When the function f is smooth (continuously differentiable) conditions on its derivatives characterise convexity (or concavity).

Proposition 2.2.8 *Let Ω be a convex set and let f be continuously differentiable on Ω. Then f is convex on Ω if and only if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})\nabla f(\mathbf{x}) \quad \forall\ \mathbf{x}, \mathbf{y} \in \Omega. \tag{2.2.12}$$

The following proposition gives sufficient conditions for f to be convex. See Appendix A for the definition and properties of positive definite matrices.

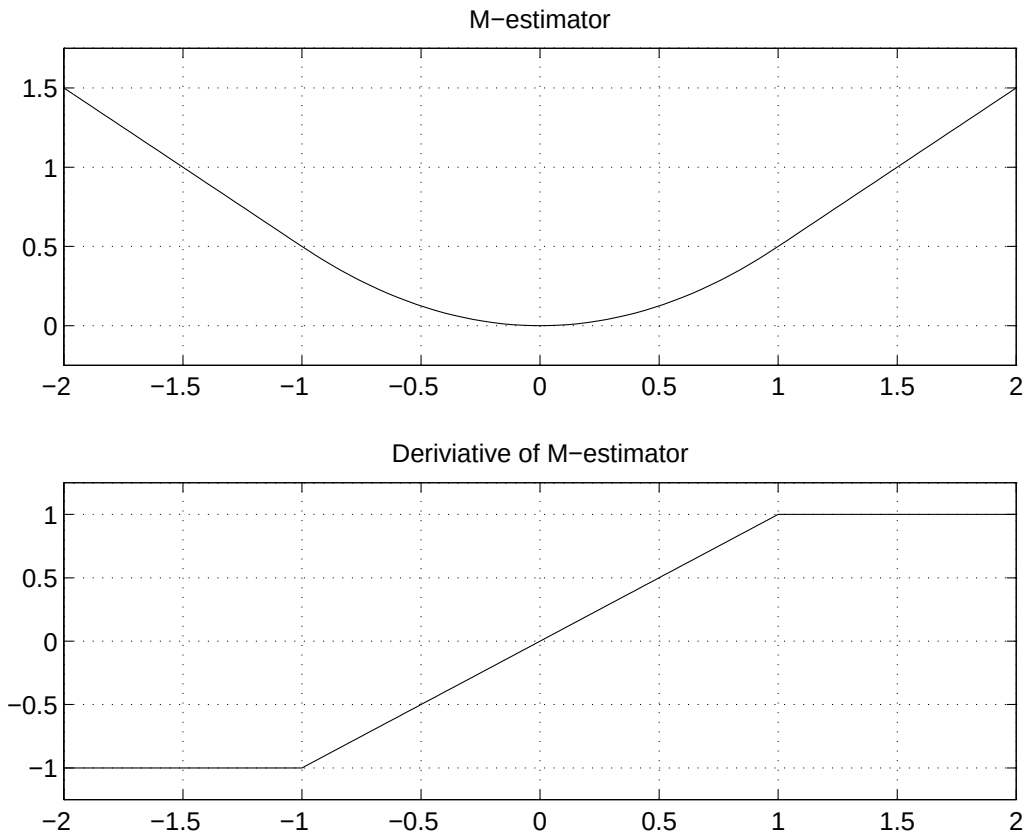


Figure 2.2.3: The M-estimator $h(t)$ for $\gamma = 1$

Proposition 2.2.9 *Let $\Omega \subseteq \mathbb{R}^n$ be convex and let $f \in C^2(\Omega)$.*

1. *If $\nabla^2 f(\mathbf{x})$ is positive definite for all $\mathbf{x} \in \Omega$ then f is strictly convex on Ω.*
2. *If $\nabla^2 f(\mathbf{x})$ is positive semi-definite for all $\mathbf{x} \in \Omega$ then f is convex on Ω.*

These conditions are sufficient for f to be convex, but they are not necessary. For example $f(x) = x^4$ on $\Omega = [-1, 1]$ is strictly convex, but $f''(x) = 12x^2$ is zero at $x = 0 \in \Omega$, so the Hessian of f is only positive semi-definite on Ω.

Lemma 2.2.10 *Let $c : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Then the set $\Omega = \{\mathbf{x} \in \mathbb{R}^n : c(\mathbf{x}) \leq 0\}$ is a convex set.*

Proof Let $\mathbf{x}, \mathbf{y} \in \Omega$ and $\theta \in [0, 1]$ be arbitrary. As $c(\mathbf{x})$ is convex

$$c(\theta\mathbf{x} + (1 - \theta\mathbf{y}) \leq \theta c(\mathbf{x}) + (1 - \theta)c(\mathbf{y}) \leq 0. \tag{2.2.13}$$

The last inequality follows as $\mathbf{x} \in \Omega$ implies $c(\mathbf{x}) \leq 0$, $\mathbf{y} \in \Omega$ implies $c(\mathbf{y}) \leq 0$ and $\theta \in [0, 1]$ implies $\theta \geq 0$ and $1 - \theta \geq 0$. Inequality (2.2.13) implies $\theta\mathbf{x} + (1 - \theta)\mathbf{y} \in \Omega$, so Ω is convex. ∎

A similar argument shows that if $c(\mathbf{x})$ is concave then the set $\{\mathbf{x} \in \mathbb{R}^n : c(\mathbf{x}) \geq 0\}$ is convex.

2.2.1 Exercises

1. Determine whether the following functions are convex, concave or neither on the given regions.

   (a) $f(\mathbf{x}) = (3x_1 - 2)^2 + 4(x_2 + 4)^2$ on $\mathbb{R}^2$.

(b) $f(\mathbf{x}) = 8x_1^2 + 2x_2^2 - 9x_1x_2$ on $\mathbb{R}^2$.

(c) $f(\mathbf{x}) = \sin(2x_1) - 3x_2^2$ on $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le \frac{\pi}{2}, -6 \le x_2 \le 7\}$.

(d) $f(\mathbf{x}) = (x_2^2 - 2x_2 + 1)e^{x_1^2}$ on $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \frac{-1}{\sqrt{2}} \le x_1 \le \frac{1}{\sqrt{2}}\}$.

2. Let f_1 and f_2 be convex functions on a convex set Ω.

   (a) i. Prove that if $\alpha_1 \ge 0$ and $\alpha_2 \ge 0$ then the function $f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \alpha_2 f_2(\mathbf{x})$ is a convex function on Ω.

   ii. Give an example of a convex set Ω, convex functions f_1, f_2 and scalars α_1 and α_2 such that $f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \alpha_2 f_2(\mathbf{x})$ is not convex.

   iii. Prove that the non-negative linear combination of a finite number of convex function is convex (Proposition 2.2.4).

   (b) i. Prove that
$$f(\mathbf{x}) = \max\{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$
   is convex on Ω.

   ii. Give an example of a convex set Ω and convex functions f_1, f_2 such that
$$\bar{f}(\mathbf{x}) = \min\{f_1(\mathbf{x}), f_2(\mathbf{x})\}$$
   is not comvex on Ω.

   iii. Prove that the maximum of a finite number of convex function is convex (Proposition 2.2.4).

3. Let Ω be a convex set in $\mathbb{R}^n$. Prove that $f : \Omega \to \mathbb{R}$ is a convex function on Ω if and only if the epigraph of f,
$$\text{epi } f = \left\{ \begin{bmatrix} \alpha \\ \mathbf{x} \end{bmatrix} \in \mathbb{R}^{n+1} : \mathbf{x} \in \Omega, \alpha \ge f(\mathbf{x}) \right\},$$
is a convex set in $\mathbb{R}^{n+1}$.

4. (a) Give an example to show that $\Omega = \{\mathbf{x} \in \mathbb{R}^n : c(\mathbf{x}) = 0\}$ may not be convex if c is nonlinear.

   (b) Give an example to show that $\Omega = \{\mathbf{x} \in \mathbb{R}^n : c(\mathbf{x}) \le 0\}$ may not be convex if c is not convex.

   (c) Prove that if the functions $c_i(\mathbf{x})$ for $i \in \mathcal{E}$ are affine, and the functions $c_i(x)$ for $i \in \mathcal{I}$ are convex then the set
$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \text{ for } i \in \mathcal{E}, \ c_i(\mathbf{x}) \le 0 \text{ for } i \in \mathcal{I}\}$$
   is convex.

5. (a) [**H**]Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Let $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ be points in $\mathbb{R}^n$ and let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be numbers satisfying
$$\sum_{i=1}^{m} \alpha_i = 1 \qquad \alpha_i \ge 0 \quad i = 1, \ldots, m.$$
   Show that
$$f\left(\sum_{i=1}^{m} \alpha_i \mathbf{x}^{(i)}\right) \le \sum_{i=1}^{m} \alpha_i f(\mathbf{x}^{(i)}).$$

   (b) [**H**]Use the previous part to derive the famous arithmetic-geometric mean inequality
$$\sum_{i=1}^{n} \alpha_i a_i \ge \prod_{i=1}^{n} a_i^{\alpha_i}$$
   where the a_i are given positive numbers and the α_i are arbitrary nonnegative weights satisfying $\sum_{i=1}^{n} \alpha_i = 1$.

6. [**H**] Let $\Omega \subseteq \mathbb{R}^n$ be a convex set.

   (a) Let $f(\mathbf{x})$ be a real valued convex function defined on Ω. Let $g(\beta)$ be a *nondecreasing* convex function defined on a real interval J where the range of $f(x)$ is contained in J. Prove that $h(\mathbf{x}) = g(f(\mathbf{x}))$ is convex.

   (b) Prove that if $f(\mathbf{x})$ is a positive concave function defined on Ω then $\frac{1}{f(\mathbf{x})}$ is convex.

   (c) Prove that if $f(\mathbf{x})$ is a convex function defined on Ω then $e^{f(\mathbf{x})}$ is convex.

2.3 Convex programming problems

Convexity of the feasible region and the objective functions provides additional information about the location of global extrema.

1. In a *convex programming problem* where both

   - the feasible region Ω is convex, and
   - the objective function $f(\mathbf{x})$ is *convex*

   then

   (a) Any local minimizer of f on Ω is a global minimizer of f on Ω.

   (b) If a global maximizer of f on Ω exists then it occurs at an extreme point of Ω.

2. In a *concave programming problem* where

   - the feasible region Ω is convex, and
   - the objective function $f(\mathbf{x})$ is *concave*

   then

   (a) Any local maximizer of f on Ω is a global maximizer of f on Ω.

   (b) If a global minimizer of f on Ω exists then it occurs at an extreme point of Ω.

For a convex programming problem, with a *strictly* convex objective, then any stationary point in the feasible region Ω is the *unique* local and global minimizer.

A strictly convex programming problem may have more than one local maxima (at different extreme points of the feasible region). Similarly for a concave programming problem, with a *strictly* concave objective, then any stationary point is the *unique* local and global maximizer. A strictly concave programming problem may have more than one local minima (at different extreme points of the feasible region).

Example 2.3.1 *Find the global maximum, if it exists, of*

$$f(\mathbf{x}) = 6x_2^4 + x_1^3 + x_1^2 - x_1x_2 + 2x_2^2 - 4x_1 + 3$$

on

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, |x_2| \le 2 \right\}.$$

The feasible region, sketched in Figure 2.3.1, is closed (as the boundary is included in Ω) and bounded (as $\|\mathbf{x}\| \le \sqrt{5}\ \forall\ \mathbf{x} \in \Omega$). As f is continuous on $\mathbb{R}^n$ global extrema exist for this problem.

The feasible region Ω is convex as it is the intersection of half-spaces defined by linear inequalities. The objective function has

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 3x_1^2 + 2x_1 - x_2 - 4 \\ 24x_2^3 - x_1 + 4x_2 \end{bmatrix} \quad \text{and} \quad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 6x_1 + 2 & -1 \\ -1 & 72x_2^2 + 4 \end{bmatrix}.$$

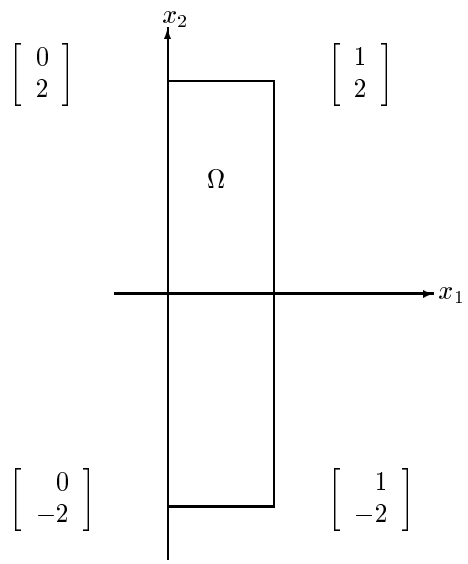


Figure 2.3.1: Feasible region Ω

For any $\mathbf{x} \in \Omega$, $x_1 \geq 0$ so

$$\begin{aligned} \operatorname{trace} \nabla^2 f(\mathbf{x}) &= 6x_1 + 72x_2^2 + 6 \geq 6, \\ \det \nabla^2 f(\mathbf{x}) &= (6x_1 + 2)(72x_2^2 + 4) - 1 \geq 144x_2^2 + 7 \geq 7. \end{aligned}$$

Thus $\nabla^2 f(\mathbf{x})$ is positive definite for all $\mathbf{x} \in \Omega$, so f is strictly convex on Ω.

As f is strictly convex on the convex set Ω

1. The global maximum occurs at an extreme point of Ω
2. Any local minimizer of f on Ω is the unique global minimizer.

The extreme points of Ω are

$$\mathbf{x}^{(a)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \mathbf{x}^{(b)} = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \quad \mathbf{x}^{(c)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \mathbf{x}^{(d)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$f(\mathbf{x}^{(a)}) = 107 \quad f(\mathbf{x}^{(b)}) = 107 \quad f(\mathbf{x}^{(c)}) = 103 \quad f(\mathbf{x}^{(d)}) = 107$$

Hence the global maximum of f on Ω is 107, which is attained at the global maximizers $\mathbf{x}^{(a)}$, $\mathbf{x}^{(b)}$ and $\mathbf{x}^{(d)}$.

Finding the unique local and global minimizer for this problem requires the optimality conditions covered in the next chapter. (The global minimizer is

$$\mathbf{x}^* = \begin{bmatrix} 0.6170868420 \\ -0.2640441415 \end{bmatrix} \quad \text{with} \quad f(\mathbf{x}^*) = 1.478974583.$$

See Example 3.1.9.)

2.3.1 Exercises

1. (a) Show that the set of minimizers of a convex programming problem is convex.
   (b) Show that a convex programming problem with a strictly convex objective function has at most one minimizer.

(c) Give an example of a convex programming problem with a strictly convex objective function which has neither a local nor global minimizer.

2. (a) Consider the problem of finding the global extrema of

$$f(\mathbf{x}) = x_1^2 - 2x_1x_2 + \frac{3}{2}x_2^2$$

on the feasible region

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_\infty \leq 1 \right\}.$$

i. Can you guarantee global extrema exist for this problem?

ii. Is this a convex programming problem?

iii. Find, if they exist, the global minimizer(s) and global maximizer(s) of $f(\mathbf{x})$ on Ω.

(b) Let $G \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. Say as much as you can about the global extrema of

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G\mathbf{x} \quad \text{on} \quad \Omega = \left\{ \mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty \leq 1 \right\}.$$

Chapter 3

Optimality Conditions

This chapter considers the *optimality conditions* which can be used to help recognize and classify local extrema. The basic problem is

$$\begin{array}{c} Minimize \quad f(\mathbf{x}) \\ \mathbf{x} \in \Omega \end{array} \tag{3.0.1}$$

where the feasible region $\Omega \subseteq \mathbb{R}^n$. The cases considered are

1. The *Unconstrained* problem where $\Omega = \mathbb{R}^n$.
2. The *Equality constrained* problem where $\mathcal{I} = \emptyset$ so
$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \ i \in \mathcal{E}\}.$$
3. The general *Inequality Constrained* problem where
$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \ i \in \mathcal{E}, \ c_i(\mathbf{x}) \leq 0 \ i \in \mathcal{I}\}.$$
4. The *Linearly Constrained* problem where
$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i^T\mathbf{x} - \beta_i = 0 \ i \in \mathcal{E}, \ \mathbf{a}_i^T\mathbf{x} - \beta_i \leq 0 \ i \in \mathcal{I}\}.$$

In the unconstrained case it is purely the curvature of the objective function $f(\mathbf{x})$ that determines a minimizer or maximizer. In the linearly constrained case both the geometry of the feasible region and the curvature of the objective function can determine local extrema. In the nonlinearly constrained case the curvature of the constraints can also play an important role. When inequality constrains are present a combinatorial aspect is introduced to the problem - which of the inequality constraints are active or binding at the solution.

The key tool used to prove the following results is a multi-variable Taylor series expansion. If $f \in C^1(\mathbb{R}^n)$ then first order Taylor series expansion of f about a point $\mathbf{x}^*$ is

$$f(\mathbf{x}^* + \alpha\mathbf{d}) = f(\mathbf{x}^*) + \alpha\mathbf{d}^T\nabla f(\mathbf{x}^*) + o(\alpha\|\mathbf{d}\|). \tag{3.0.2}$$

If $f \in C^2(\mathbb{R}^n)$ then the second order Taylor series expansion of f about a point $\mathbf{x}^*$ is

$$f(\mathbf{x}^* + \alpha\mathbf{d}) = f(\mathbf{x}^*) + \alpha\mathbf{d}^T\nabla f(\mathbf{x}^*) + \tfrac{1}{2}\alpha^2\mathbf{d}^T\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha^2\|\mathbf{d}\|^2). \tag{3.0.3}$$

3.1 Unconstrained Problems

3.1.1 First order necessary conditions

Theorem 3.1.1 (First order necessary conditions) *Let $\Omega = \mathbb{R}^n$ and $f \in C^1(\mathbb{R}^n)$. If $\mathbf{x}^*$ a local minimizer of f then $\nabla f(\mathbf{x}^*) = 0$.*

Proof As $\mathbf{x}^*$ is a local minimizer there exits an $\bar{\alpha} > 0$ such that $f(\mathbf{x}^* + \alpha\mathbf{d}) \geq f(\mathbf{x}^*)$ for all $\alpha \in [0, \bar{\alpha}]$ and for all $\mathbf{d}$ with $\|\mathbf{d}\| = 1$. A first order Taylor series expansion gives

$$0 \leq f(\mathbf{x}^* + \alpha\mathbf{d}) - f(\mathbf{x}^*) = \alpha\mathbf{d}^T\nabla f(\mathbf{x}^*) + o(\alpha) \quad \forall \mathbf{d} \in \mathbb{R}^n, \|\mathbf{d}\| = 1.$$

Dividing by α and letting $\alpha \to 0^+$ produces

$$0 \leq \mathbf{d}^T\nabla f(\mathbf{x}^*) \quad \forall \mathbf{d} \in \mathbb{R}^n, \|\mathbf{d}\| = 1.$$

Taking $\mathbf{d} = \pm\mathbf{e}_i$, where $\mathbf{e}_i$ is the i-th unit vector in $\mathbb{R}^n$, shows that $\nabla f(\mathbf{x}^*) = 0$. ∎

A similar argument also shows that if $\mathbf{x}^*$ is an unconstrained local maximizer then $\nabla f(\mathbf{x}^*) = 0$.

Definition 3.1.2 *If $\Omega = \mathbb{R}^n$ and $f \in C^1(\mathbb{R}^n)$ then any point $\mathbf{x}^*$ with $\nabla f(\mathbf{x}^*) = 0$ is an* unconstrained stationary point.

Definition 3.1.3 *A stationary point $\mathbf{x}^* \in \Omega$ is a* saddle point *of f if for any $\delta > 0$ there exist $\mathbf{x}, \mathbf{y}$ with $\|\mathbf{x} - \mathbf{x}^*\| < \delta$, $\|\mathbf{y} - \mathbf{x}^*\| < \delta$ such that $f(\mathbf{x}) < f(\mathbf{x}^*)$ and $f(\mathbf{y}) > f(\mathbf{x}^*)$.*

That is in any neighbourhood of a saddle point $\mathbf{x}^*$ there are points with function value less that $f(\mathbf{x}^*)$ and points with function value greater than $f(\mathbf{x}^*)$.

An unconstrained stationary point can be either a minimizer, a maximizer or a saddle point. More information on the nature of the stationary point can be obtained from considering second derivative information.

3.1.2 Second order conditions

Theorem 3.1.4 (Second order necessary conditions) *Let $\Omega = \mathbb{R}^n$ and let $f \in C^2(\mathbb{R}^n)$. If $\mathbf{x}^*$ is a local minimizer of f then $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite.*

Proof As $\mathbf{x}^*$ is a local minimizer there exists an $\bar{\alpha} > 0$ such that

$$f(\mathbf{x}^* + \alpha\mathbf{d}) \geq f(\mathbf{x}^*) \quad \forall \alpha \in [0, \bar{\alpha}] \quad \forall \mathbf{d} : \|\mathbf{d}\| = 1.$$

A second order Taylor series expansion of f about $\mathbf{x}^*$ gives

$$0 \leq f(\mathbf{x}^* + \alpha\mathbf{d}) - f(\mathbf{x}^*) = \frac{\alpha^2}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x}^*)\mathbf{d} + o(\alpha^2).$$

Dividing by α^2 and letting $\alpha \to 0^+$ shows that

$$0 \leq \mathbf{d}^T\nabla^2 f(\mathbf{x}^*)\mathbf{d} \quad \forall \mathbf{d} : \|\mathbf{d}\| = 1$$

so $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite. ∎

Similarly it can be established that if $\mathbf{x}^*$ is a local maximizer then $\nabla^2 f(\mathbf{x}^*)$ is negative semi-definite. These necessary conditions do not guarantee a point is a local minimizer or a local maximizer. However any point which does not satisfy the necessary conditions for a local minimizer cannot be a local minimizer. The contrapositive of the necessary conditions Theorem 3.1.4 gives

Corollary 3.1.5 *Let $\mathbf{x}^*$ be a stationary point of $f \in C^2(\mathbb{R}^n)$, so $\nabla f(\mathbf{x}^*) = 0$. Then*

1. *$\nabla^2 f(\mathbf{x}^*)$ has a negative eigenvalue $\Longrightarrow$ $\mathbf{x}^*$ is **not** a strict local minimizer.*
2. *$\nabla^2 f(\mathbf{x}^*)$ has a positive eigenvalue $\Longrightarrow$ $\mathbf{x}^*$ is **not** a strict local maximizer.*
3. *$\nabla^2 f(\mathbf{x}^*)$ indefinite (i.e. it has both a positive eigenvalue and a negative eigenvalue) $\Longrightarrow$ $\mathbf{x}^*$ is neither a local minimizer nor a local maximizer.*

Theorem 3.1.6 (Second order sufficient conditions) *Let $\Omega = \mathbb{R}^n$ and let $f \in C^2(\mathbb{R}^n)$. If $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is positive definite then $\mathbf{x}^*$ is a strict local minimizer.*

Proof Let $\mathbf{x}^*$ be a point where $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is positive definite. For a proof by contradiction assume that $\mathbf{x}^*$ is *not* a strict local minimizer. Then there exists a sequence $\{\mathbf{d}^{(k)}\}$ such that $\mathbf{d}^{(k)} \to 0$ and $f(\mathbf{x}^* + \mathbf{d}^{(k)}) \le f(\mathbf{x}^*)$ for all $k \ge \bar{k}$. As $\mathbf{x}^*$ is a stationary point a second order Taylor series expansion gives

$$0 \ge f(\mathbf{x}^* + \mathbf{d}^{(k)}) - f(\mathbf{x}^*) = \tfrac{1}{2}\mathbf{d}^{(k)^T}\nabla^2 f(\mathbf{x}^*)\mathbf{d}^{(k)} + o(\|\mathbf{d}^{(k)}\|^2) \quad \forall\ k \ge \bar{k}.$$

As $\nabla^2 f(\mathbf{x}^*)$ is positive definite there exists an $\eta^* > 0$ (the smallest eigenvalue of $\nabla^2 f(\mathbf{x}^*)$) such that $\mathbf{d}^T\nabla^2 f(\mathbf{x}^*)\mathbf{d} \ge \eta^*\mathbf{d}^T\mathbf{d}$ for all $\mathbf{d}$. Hence

$$0 \ge \frac{\eta^*}{2}\mathbf{d}^{(k)^T}\mathbf{d}^{(k)} + o(\|\mathbf{d}^{(k)}\|^2).$$

Dividing by $\|\mathbf{d}^{(k)}\|^2$ and letting $k \to \infty$ gives

$$0 \ge \frac{\eta^*}{2} > 0$$

which is a contradiction. ∎

If $\mathbf{x}^*$ is a stationary point with $\nabla^2 f(\mathbf{x}^*)$ negative definite then a similar argument shows that $\mathbf{x}^*$ is a strict local maximizer. In summary, if $\mathbf{x}^*$ is an unconstrained ($\Omega = \mathbb{R}^n$) stationary point, so $\nabla f(\mathbf{x}^*) = 0$, then

1. $\nabla^2 f(\mathbf{x}^*)$ positive definite $\Longrightarrow$ $\mathbf{x}^*$ is a strict local minimizer.
2. $\nabla^2 f(\mathbf{x}^*)$ negative definite $\Longrightarrow$ $\mathbf{x}^*$ is a strict local maximizer.

If $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite or negative semi-definite, with a zero eigenvalue, then higher order derivatives are needed to determine if $\mathbf{x}^*$ is a local minimizer, local maximizer or neither. Some information can be gained from Corollary 3.1.5.

If global information (convexity or concavity) about f is known then the nature of a stationary point is also known.

1. If f is convex on $\mathbb{R}^n$ then any stationary point is a global minimizer.
2. If f is strictly convex on $\mathbb{R}^n$ then any stationary point is the unique global minimizer.
3. If f is concave on $\mathbb{R}^n$ then any stationary point is a global maximizer.
4. If f is strictly concave on $\mathbb{R}^n$ then any stationary point is the unique global maximizer.

Example 3.1.7 (UCEX1) *Consider the function* $f(\mathbf{x}) = x_1^2 - 2x_1^3 - x_2^3$.

1. *Show that*
$$\overline{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textit{and} \quad \hat{\mathbf{x}} = \begin{bmatrix} \frac{1}{3} \\ 0 \end{bmatrix} \tag{3.1.1}$$
are stationary points of $f(\mathbf{x})$.
2. *Using second order information determine, if possible, the nature of these stationary points.*
3. *Show that* $\overline{\mathbf{x}}$ *satisfies the second order necessary conditions for a minimizer. However, by looking at the function value along a line through* $\overline{\mathbf{x}}$, *show that* $\overline{\mathbf{x}}$ *is not a local minimizer.*

The gradient and Hessian of f are

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 - 6x_1^2 \\ -3x_2^2 \end{bmatrix} \qquad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 - 12x_1 & 0 \\ 0 & -6x_2 \end{bmatrix}. \tag{3.1.2}$$

Substituting (3.1.1) in (3.1.2) shows that $\nabla f(\overline{\mathbf{x}}) = 0$ and $\nabla f(\hat{\mathbf{x}}) = 0$. In fact these are the only stationary points as any stationary points must satisfy $2x_1(1 - 3x_1) = 0$ and $-3x_2^2 = 0$, so $x_2 = 0$ and either $x_1 = 0$ or $x_1 = \frac{1}{3}$.

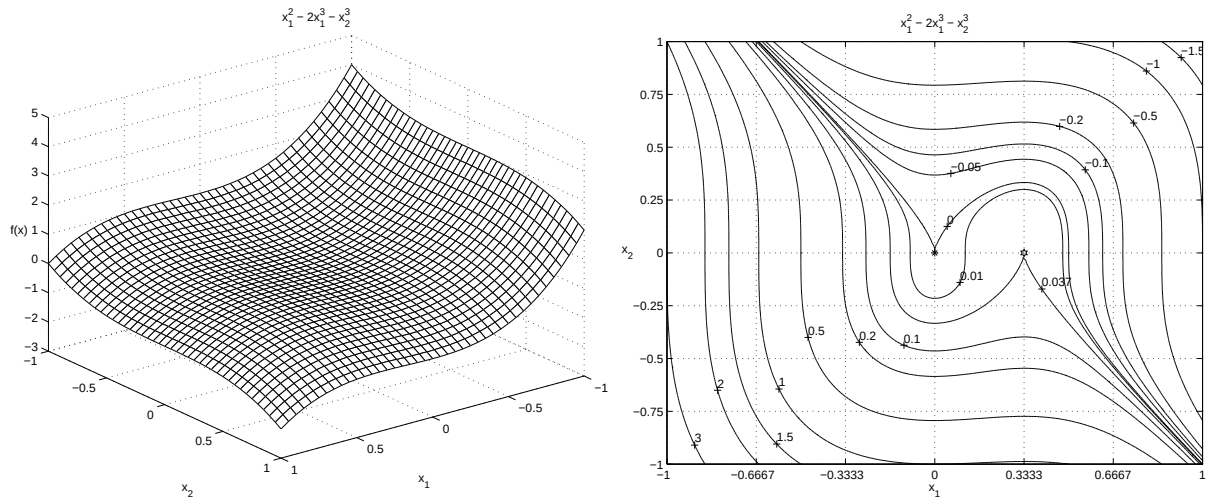


Figure 3.1.1: Surface and contour plots of $f(\mathbf{x}) = x_1^2 - 2x_1^3 - x_2^3$

At $\bar{\mathbf{x}}$ the Hessian

$$\nabla^2 f(\bar{\mathbf{x}}) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

is positive semi-definite (eigenvalues are 2 and 0). Thus $\bar{\mathbf{x}}$ satisfies the second order necessary conditions for a minimum, but not the second order sufficient conditions for a minimum. From the Hessian it is only possible to say that $\bar{\mathbf{x}}$ is *not* a local maximizer.

At $\hat{\mathbf{x}}$ the Hessian

$$\nabla^2 f(\hat{\mathbf{x}}) = \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}.$$

is negative semi-definite. Thus $\hat{\mathbf{x}}$ satisfies the second order necessary conditions for a maximum, but not the second order sufficient conditions for a maximum. From the Hessian it is only possible to say that $\hat{\mathbf{x}}$ is *not* a local minimizer.

Sometimes it is possible to get more information about the nature of a stationary point by looking at the function value along a line passing through the point. If

$$\mathbf{d} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad f(\bar{\mathbf{x}} + \alpha\mathbf{d}) = -\alpha^3$$

so $\bar{\mathbf{x}}$, which corresponds to $\alpha = 0$, is a saddle point as the function value both increases for $\alpha < 0$ and decreases for $\alpha > 0$. The fact that the function value along a line through $\bar{\mathbf{x}}$ has a minimum at $\alpha = 0$, for example

$$\mathbf{d} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad f(\bar{\mathbf{x}} + \alpha\mathbf{d}) = \alpha^2 - 2\alpha^3$$

does not imply that $\bar{\mathbf{x}}$ is a local minimizer.

Similarly looking at the function value along lines passing through $\hat{\mathbf{x}}$ shows that $\hat{\mathbf{x}}$ is a saddle point. The surface and contour plots of f are sketched in Figure 3.1.1. A `Maple` text file for this example is in the file `ucex1map.txt`.

Example 3.1.8 (UCEX2) *Consider the function* $f(\mathbf{x}) = x_1^4 - 3x_1^3 + 4x_1^2 - 6x_1x_2 + 3x_2^2$.

1. *Is the point* $\mathbf{y} = [\,1 \quad 1\,]^T$ *a stationary point of* $f(\mathbf{x})$*?*
2. *Find all stationary points of* $f(\mathbf{x})$.
3. *Identify, as far as possible using second order information, the nature of all the stationary points (i.e. (strict) local minimizer, (strict) local maximizer, saddle point).*

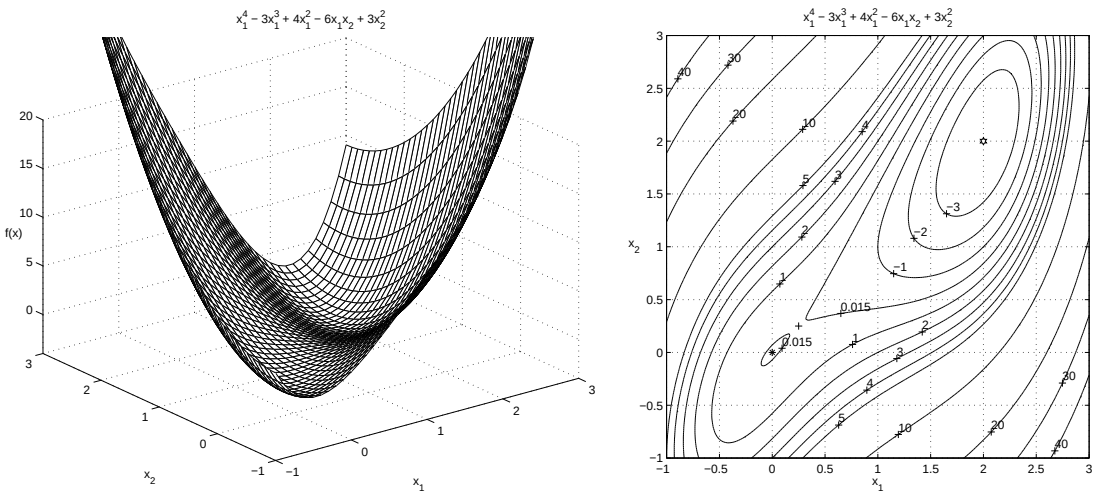


Figure 3.1.2: Surface and contour plots of $f(\mathbf{x}) = x_1^4 - 3x_1^3 + 4x_1^2 - 6x_1x_2 + 3x_2^2$

The gradient and Hessian of f are

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 4x_1^3 - 9x_1^2 + 8x_1 - 6x_2 \\ -6x_1 + 6x_2 \end{bmatrix} \qquad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 12x_1^2 - 18x_1 + 8 & -6 \\ -6 & 6 \end{bmatrix}.$$

The point

$$\mathbf{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Longrightarrow \nabla f(\mathbf{y}) = \begin{bmatrix} -3 \\ 0 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

so $\mathbf{y}$ is not a stationary point, so it is neither a local minimizer, nor a local maximizer nor a saddle point.

A stationary point $\nabla f(\mathbf{x}) = 0$ has $x_1 = x_2$ and $4x_1^3 - 9x_1^2 + 8x_1 - 6x_2 = x_1(4x_1^2 - 9x_1 + 2) = x_1(x_1 - 2)(4x_1 - 1) = 0$. Thus the stationary points are

$$\bar{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \hat{\mathbf{x}} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad \tilde{\mathbf{x}} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}.$$

At $\bar{\mathbf{x}}$ the Hessian

$$\nabla^2 f(\bar{\mathbf{x}}) = \begin{bmatrix} 8 & -6 \\ -6 & 6 \end{bmatrix},$$

is positive definite as $\operatorname{trace} \nabla^2 f(\bar{\mathbf{x}}) = 14 > 0$ and $\det \nabla^2 f(\bar{\mathbf{x}}) = 12 > 0$. Hence $\bar{\mathbf{x}}$ is a strict local minimizer with $f(\bar{\mathbf{x}}) = 0$.

At $\hat{\mathbf{x}}$ the Hessian

$$\nabla^2 f(\hat{\mathbf{x}}) = \begin{bmatrix} 20 & -6 \\ -6 & 6 \end{bmatrix},$$

is positive definite as $\det \nabla^2 f(\hat{\mathbf{x}}) = 84 > 0$ and $\operatorname{trace} \nabla^2 (f(\hat{\mathbf{x}}) = 26 > 0$. Hence $\hat{\mathbf{x}}$ is also a strict local minimizer with $f(\hat{\mathbf{x}}) = -4$.

At $\tilde{\mathbf{x}}$ the Hessian

$$\nabla^2 f(\tilde{\mathbf{x}}) = \begin{bmatrix} \frac{17}{4} & -6 \\ -6 & 6 \end{bmatrix},$$

is indefinite as $\det \nabla^2 f(\tilde{\mathbf{x}}) = -\frac{21}{2} < 0$. Hence $\tilde{\mathbf{x}}$ is a saddle point with $f(\tilde{\mathbf{x}}) = \frac{5}{256}$.

The surface and contour plots of f are sketched in Figure 3.1.2. A `Maple` text file for this example is in the file `ucex2map.txt`.

Example 3.1.9 (UCEX3) *Consider the problem in Example 2.3.1 where*

$$f(\mathbf{x}) = 6x_2^4 + x_1^3 + x_1^2 - x_1x_2 + 2x_2^2 - 4x_1 + 3$$

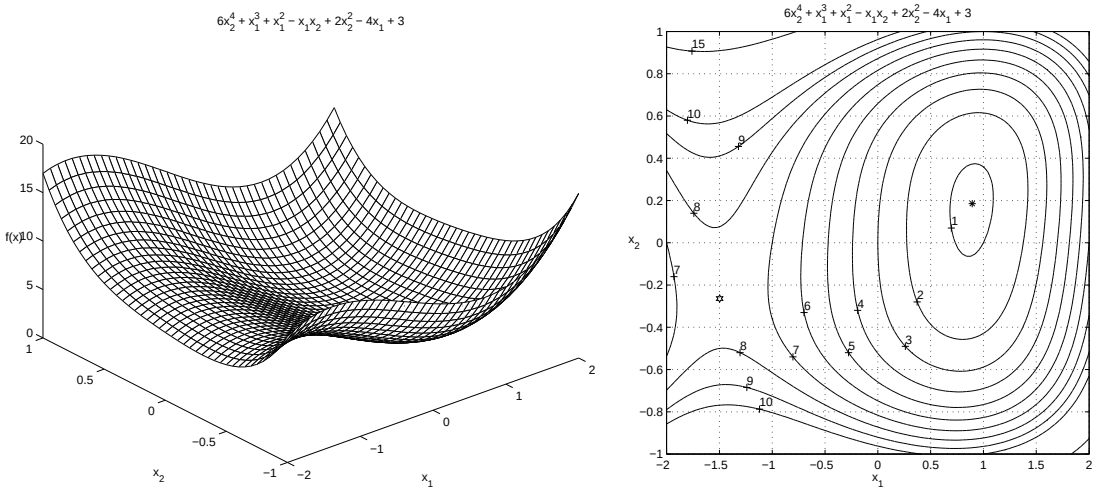


Figure 3.1.3: Surface and contour plots of $f(\mathbf{x}) = 6x_2^4 + x_1^3 + x_1^2 - x_1x_2 + 2x_2^2 - 4x_1 + 3$

and

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^2 : 0 \le x_1 \le 1, |x_2| \le 2 \right\}.$$

This problem is plotted in Figure 3.1.3. In Example 2.3.1 it was established that Ω is convex and f is strictly convex on Ω, so any local minimizer of f on Ω is the strict global minimizer. Moreover if f has a stationary point $\mathbf{x}^* \in \Omega$ then convexity implies this is the unique local and global minimizer.

1. *Find all the stationary points of $f(\mathbf{x})$.*
2. *Show that $f(\mathbf{x})$ has a stationary point, which is an unconstrained saddle point, but does not lie in the feasible region Ω.*
3. *Show that $f(\mathbf{x})$ has a stationary point $\mathbf{x}^*$ in the interior of the feasible region Ω, and that $\mathbf{x}^*$ is the global minimizer of $f(\mathbf{x})$ on Ω.*

A stationary point satisfies

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 3x_1^2 + 2x_1 - x_2 - 4 \\ 24x_2^3 - x_1 + 4x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The second equation implies $x_1 = 24x_2^3 + 4x_2$. Substituting in the first equations gives

$$\phi(x_2) = 1728x_2^6 + 576x_2^4 + 48x_2^2 + 48x_2^3 + 7x_2 - 4 = 0$$

The function $\phi(x_2)$ is plotted in Figure 3.1.4, from which it is clear that ϕ has two roots, one in the interval $(-0.3, -0.2)$ and the other in the interval $(0.1, 0.2)$. Solving the equation $\phi(x_2) = 0$ numerically gives $\bar{x}_2 = -0.2640441415$ and $x_2^* = 0.1853071230$. Using $x_1 = 24x_2^3 + 4x_2$ the first root $\bar{x}_2$ gives the stationary point

$$\bar{\mathbf{x}} = \begin{bmatrix} -1.497991966 \\ -0.2640441415 \end{bmatrix}.$$

The point $\overline{\mathbf{x}}$ is not in the feasible region as $\bar{x}_1 < 0$. If there were no constraints present the point $\overline{\mathbf{x}}$ is an unconstrained saddle point of $f(\mathbf{x})$ with $f(\overline{\mathbf{x}}) = 7.647551194$. The positive root x_2^* gives the stationary point

$$\mathbf{x}^* = \begin{bmatrix} 0.8939455614 \\ 0.1853071230 \end{bmatrix} \quad \text{with} \quad f(\mathbf{x}^*) = 0.8478407550.$$

As $\mathbf{x}^* \in \Omega$ the convexity of both Ω and $f(\mathbf{x})$ implies that $\mathbf{x}^*$ is the unique local and global minimizer of f on Ω.

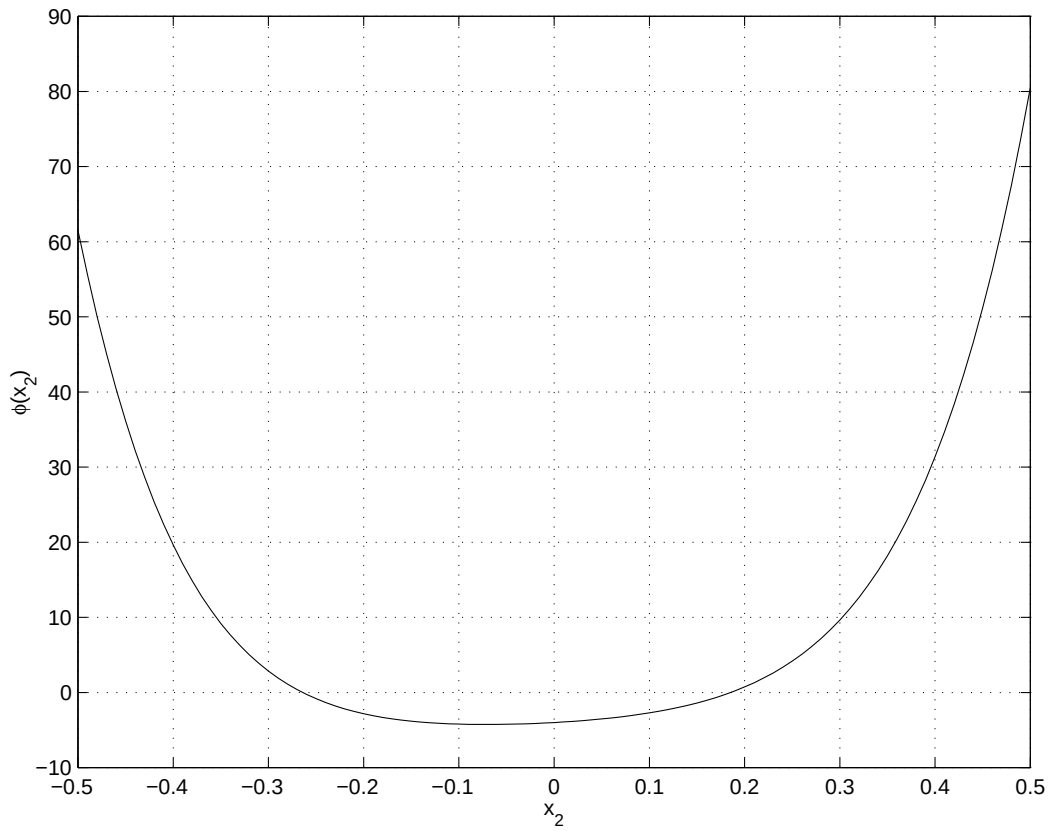


Figure 3.1.4: Plot of $\phi(x_2) = 1728x_2^6 + 576x_2^4 + 48x_2^2 + 48x_2^3 + 7x_2 - 4$

If it was not known that this is a convex programming problem then, as $\mathbf{x}^*$ is in the interior of $\mathbf{x}^*$ (none of the constraints defining Ω are satisfied as equalities) the Hessian of $f(\mathbf{x})$ may provide some information about the nature of $\mathbf{x}^*$. Evaluating the Hessian gives

$$\nabla^2 f(\mathbf{x}^*) = \begin{bmatrix} 7.363673370 & -1 \\ -1 & 6.4722388548 \end{bmatrix},$$

which has eigenvalues 5.82 and 8.01, so $\nabla^2 f(\mathbf{x}^*)$ is positive definite. As the stationary point $\mathbf{x}^*$ is in the interior of Ω has a positive definite Hessian it is a strict local minimizer of f on Ω.

A `Maple` text file for this example is in the file `ucex3map.txt`.

3.1.3 Exercises

1. Find all the stationary points of the following functions and identify, if possible, the local minimizers, local maximizers and saddle points.

   (a) $f(\mathbf{x}) = x_1^2 + 3x_2^2 - 3x_1x_2 + 6$

   (b) $f(\mathbf{x}) = x_2^2 \sin^2(x_1 - 1)$

   (c) $f(\mathbf{x}) = 2x_1^3x_2^2 - 4x_1^3 + 3x_1^2x_2^2 - 6x_1^2$

   (d) $f(\mathbf{x}) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$

2. Investigate whether the origin in $\mathbb{R}^3$ is an extrema of the function

$$f(\mathbf{x}) = \alpha x_1^2 e^{x_2} + x_2^2 e^{x_3} + x_3^2 e^{x_1},$$

   where $\alpha \in \mathbb{R}$ is a parameter.

3. Show that the function $f(\mathbf{x}) = (x_2 - x_1^2)^2 + x_1^5$ has only one stationary point $x \in \mathbb{R}^2$, and that this stationary point is neither a local minimizer nor a local maximizer.

4. Consider the function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\mathbf{x}) = x_1^2 + 2x_1x_2 + x_2^2 + \alpha x_1 + \beta x_2,$$

where α and β are real parameters.

   (a) Show that if $\alpha \neq \beta$ then this function has no stationary points.

   (b) Find all the stationary points when $\alpha = \beta$.

   (c) Show that f is a convex function on $\mathbb{R}^2$, so that all the stationary points found in (b) are global minimizers of f. Verify the the set of global minimizers of f is a convex set.

   (d) Sketch the contours of f when $\alpha = \beta = 4$, and mark the global minimizers on your sketch.

5. Rosenbrock's function is $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

   (a) Show that $\mathbf{x}^* = [\ 1 \quad 1\ ]^T$ satisfies $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ positive definite.

   (b) [H] Show that $\nabla^2 f(x)$ is singular if and only if $x_2 - x_1^2 = 0.005$. Hence show that $\nabla^2 f(x)$ is positive definite for all $x \in \mathbb{R}^2$ such that $f(x) < 0.0025$.

6. **[H]** Let $p, q \in \mathbb{R}$ satisfy $0 < p < q$, and let

$$f(\mathbf{x}) = (x_2 - px_1^2)(x_2 - qx_1^2).$$

   (a) Show that the origin in $\mathbb{R}^2$ is a local minimizer of f along every line $\mathbf{x}(\alpha) = \alpha\mathbf{d}$ passing through the origin. Here $\mathbf{d} \in \mathbb{R}^2$ is a non-zero vector giving the direction of the line.

   (b) Let $\mathbf{x}(\alpha) = [\ \alpha \quad \epsilon\alpha^2\ ]^T$. Show that if $p < \epsilon < q$ and $\alpha \neq 0$ then $f(\mathbf{x}(\alpha)) < 0$, while $f(\mathbf{x}(\alpha)) \geq 0$ if $\epsilon \leq p$ or $\epsilon \geq q$.

   (c) Show that the origin satisfies the second order necessary conditions, but is neither a local minimizer, a local maximizer nor a saddle point.

7. The "peaks" function from MATLAB is

$$f(\mathbf{x}) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{-x_1^2 - x_2^2} - \frac{1}{3} e^{-(x_1+1)^2 - x_2^2}$$

on the feasible region $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_\infty \leq 3\}$. Figure 3.1.5 gives a surface plot of $f(\mathbf{x})$ on Ω.

   (a) Can you be sure both a global minimum and a global maximum exists for this problem?

   (b) A student is having difficulty evaluating the gradient $\nabla f(\mathbf{x})$ and the Hessian $\nabla^2 f(\mathbf{x})$. Suggest ways expressions for the gradient/Hessian can be calculated and/or verified.

   (c) A plot of the peaks function is given in Figure 3.1.5. From this it looks like there are three local maxima, two local minima, and several stationary points, all in the interior of Ω.

   Find as many of the stationary points of $f(\mathbf{x})$ as you can, in each case determining if it is a local minimizer, local maximizer or saddle point.

   (d) Accurately find the global minimum and global maximum of $f(\mathbf{x})$ on Ω.

Notes

- The `Maple` text files `ucex1map.txt`, `ucex2map.txt` and `ucex3map.txt` in the class account, solving an examples from lectures, may be useful. One approach is to edit a text file and then use the *Import Text* item on the *File* menu. The *Export Text* item can be used to save the results of your `Maple` session for printing.

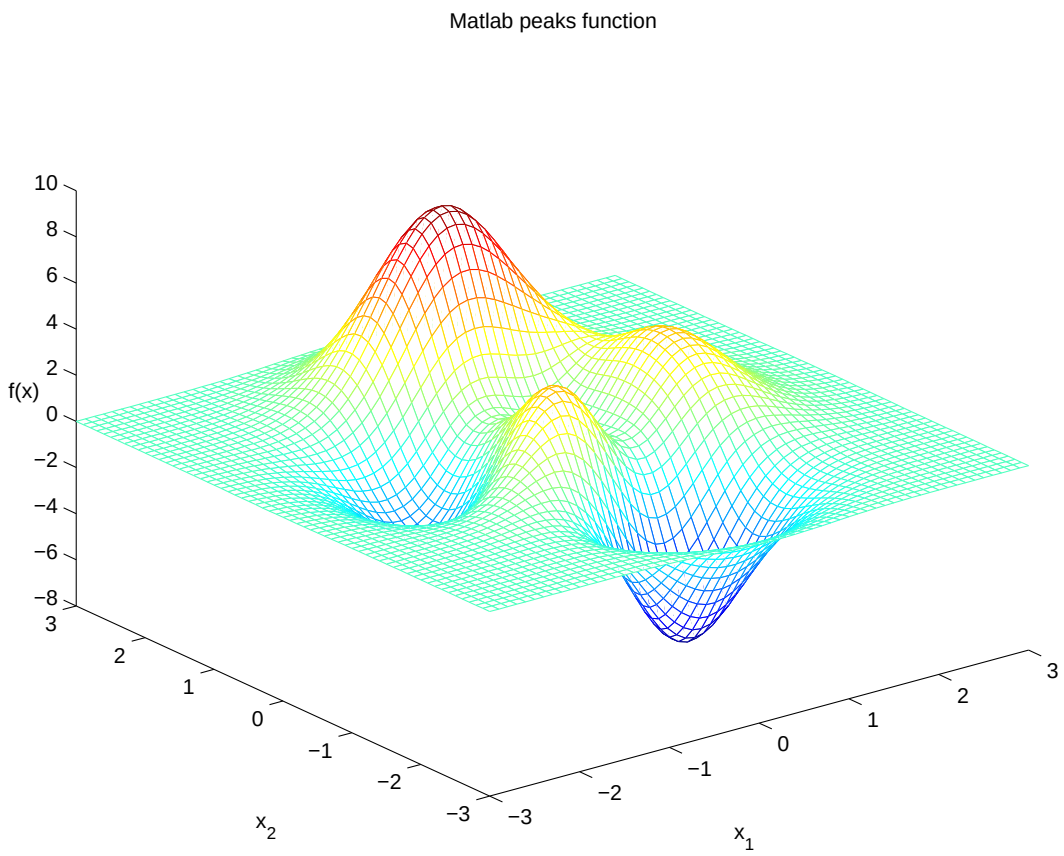


Figure 3.1.5: Plot of $f(\mathbf{x})$ on Ω

- The `Maple` command `solve` may not be able to find all/any of the stationary points. The command `fsolve` may then be useful, particularly in conjunction with specifying a small region in which the stationary point lies. Help on these commands can be obtained by the `Maple` commands `?solve` or `?fsolve`.
- A `Maple` plot may be rotated by using the left mouse button to move the frame box around. The middle mouse button will redraw the plot. Be careful of the effects of looking at the plot from an angle when estimating the coordinates of a point on the plot.

3.2 Equality Constraints

The equality constrained problem is

$$\begin{array}{ll} Minimize & f(\mathbf{x}) \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & c_i(\mathbf{x}) = 0 \quad i \in \mathcal{E} \end{array} \tag{3.2.1}$$

Let $m = |\mathcal{E}|$ be the number of equality constraints and assume the equality constraints are numbered so $\mathcal{E} = \{1, \ldots, m\}$.

The only points of interest are the feasible points $\mathbf{x} \in \Omega$. Assuming Ω is not just an isolated point then a minimum can be due either to the curvature of the objective function or to the curvature of the equality constraint(s) or a combination of both.

Example 3.2.1 (Equality constrained minimum from objective curvature) *Let*

$$f(\mathbf{x}) = x_1^2 + 6x_1x_2 + x_2^2 - 12x_1 - 4x_2 \qquad \Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1 - x_2 + 1 = 0\}.$$

A surface plot and contour plot of f are given in Figure 3.2.1. Part of the feasible region Ω, which

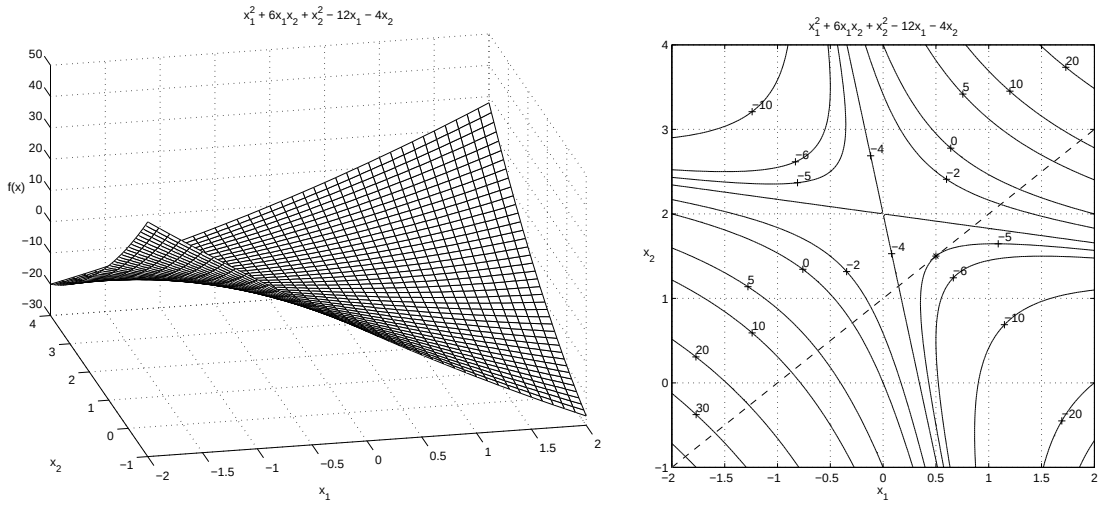


Figure 3.2.1: Surface and contour plots of $f(\mathbf{x}) = x_1^2 + 6x_1x_2 + x_2^2 - 12x_1 - 4x_2$

corresponds to the line $x_2 = x_1 + 1$, is also shown on the contour plot. The curvature of the objective function creates a minimum around $[\ \frac{1}{2} \quad \frac{3}{2}\ ]^T$. If there is no constraint present f has a stationary point at $\hat{\mathbf{x}} = [\ 0 \quad 2\ ]^T$, which is neither a minimizer nor a maximizer (see Example 3.2.6).

When there are just two variables and one constraint, the constraint can be used to eliminate one of the variables, reducing the problem to an unconstrained minimization of a function of one variable. In Example 3.2.1 this can be done by setting $x_2 = x_1 + 1$, and substituting for x_2 in the objective function to produce

$$\hat{f}(x_1) = 8x_2^2 - 8x_1 - 3.$$

The function $\hat{f}(x_1)$ has the minimizer $x_1^* = \frac{1}{2}$ (so $x_2^* = \frac{3}{2}$) with minimum value $\hat{f}(x_1^*) = f(\mathbf{x}^*) = -4$.

Example 3.2.2 (Equality constrained minimum from constraint curvature) *Let*

$$f(\mathbf{x}) = -x_1 + x_2 + 5 \qquad \Omega = \left\{\, \mathbf{x} \in \mathbb{R}^2 : -x_1^2 + x_2 = 0 \,\right\}.$$

A surface plot and contour plot of f are given in Figure 3.2.2. Part of the feasible region Ω, which is the curve $x_2 = x_1^2$, is also shown on the contour plot. The curvature of the constraint function creates a minimum around $[\ \frac{1}{2} \quad \frac{1}{4}\ ]^T$ (see Example 3.2.11). The objective function is affine, so if there were no constraints present there would not be a finite minimum or maximum. For instance if $\mathbf{x}(\alpha) = [\ \alpha \quad 0\ ]^T$, then $f(\mathbf{x}(\alpha)) \to \infty$ as $\alpha \to -\infty$ and $f(\mathbf{x}(\alpha)) \to -\infty$ as $\alpha \to +\infty$.

Again the constraint in Example 3.2.2 can be used to eliminate one of the variables by setting $x_2 = x_1^2$, reducing the problem to the unconstrained minimization of the function

$$\hat{f}(x_1) = x_1^2 - x_1 + 5.$$

This function has a minimizer at $x_1^* = \frac{1}{2}$ (so $x_2^* = \frac{1}{4}$), with minimum value $\hat{f}(x_1^*) = f(\mathbf{x}^*) = 4\frac{3}{4}$.

The ability to use equality constraints, in particular linear equality constraints, to eliminate some of the variables, so reducing the size of the problem, is one of the reasons for treating equality constraints separately.

Associated with the equality constrained problem is the *Lagrangian function*

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i c_i(\mathbf{x})$$

where there is a Lagrange multiplier λ_i corresponding to each constraint $c_i(\mathbf{x})$ for $i = 1, \ldots, m$.

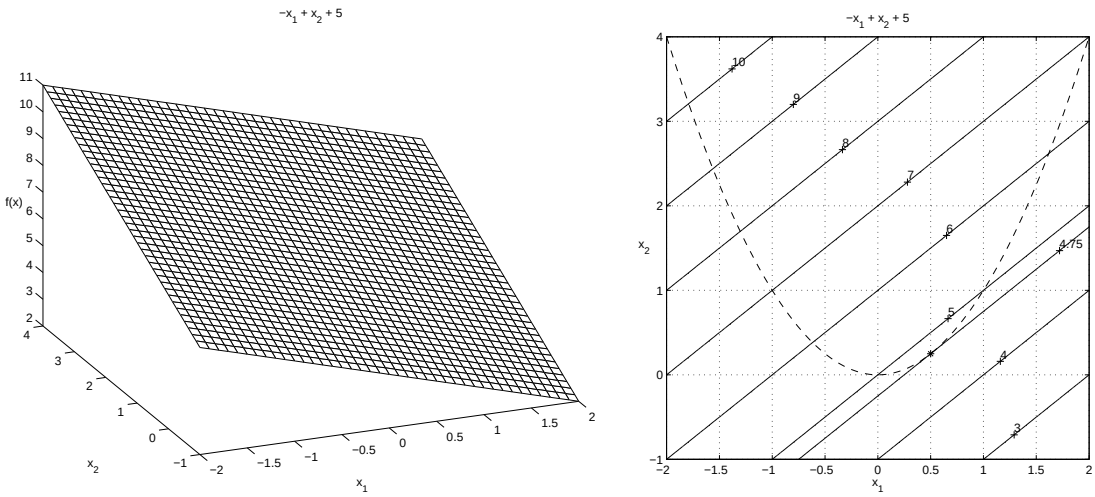


Figure 3.2.2: Surface and contour plots of $f(\mathbf{x}) = -x_1 + x_2 + 5$

3.2.1 First order necessary conditions

Necessary conditions for constrained optimization require a regularity condition on the geometry of the feasible region.

Definition 3.2.3 *A feasible point* $\bar{\mathbf{x}}$ *is a* regular point *if and only if the gradients* $\nabla c_i(\bar{\mathbf{x}}), i \in \mathcal{E}$ *are linearly independent.*

As a feasible point satisfies the m equality constraints $c_i(\mathbf{x}) = 0$ for $i = 1, \ldots, m$ it follows that $m \leq n$ at a regular point.

Let

$$A(\mathbf{x}) = \left[\begin{array}{ccc} \nabla c_1(\mathbf{x}) & \cdots & \nabla c_m(\mathbf{x}) \end{array} \right].$$

$A(x)$ is the transpose of the Jacobian matrix

$$J(\mathbf{x}) = \left[\begin{array}{c} \nabla c_1(\mathbf{x})^T \\ \vdots \\ \nabla c_m(\mathbf{x})^T \end{array} \right] = A(\mathbf{x})^T.$$

A feasible point $\mathbf{x}$ is then a regular point if and only if the n by m matrix $A(\mathbf{x})$ has full column rank, that is $\operatorname{rank} A(\mathbf{x}) = m$.

Theorem 3.2.4 (First Order Necessary Conditions) *Let* $\Omega = \{x \in \mathbb{R}^n : c_i(\mathbf{x}) = 0, i = 1, \ldots, m\}$ *and* $f \in C^1(\Omega)$. *If* $\mathbf{x}^*$ *is a local minimizer of* f *on* Ω *and if* $\mathbf{x}^*$ *is a regular point then there exist Lagrange multipliers* $\lambda_i, i = 1, \ldots, m$ *such that*

$$\nabla_x \mathcal{L}(\mathbf{x}^*, \lambda^*) = 0 \tag{3.2.2}$$

and

$$\nabla_\lambda \mathcal{L}(\mathbf{x}^*, \lambda^*) = 0 \tag{3.2.3}$$

If a local minimizer $\mathbf{x}^*$ is *not* a regular point, that is the constraint gradients $\nabla c_i(\mathbf{x}^*), i = 1, \ldots, m$ are linearly dependent, there are two possibilities

1. There do not exist multipliers $\lambda^* \in \mathbb{R}^m$ satisfying $\nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\lambda^* = 0$.

2. There are an infinite number of multipliers $\lambda^* \in \mathbb{R}^m$ satisfying $\nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\lambda^* = 0$.

A regular point guarantees the *existence* and *uniqueness* of the multipliers satisfying $\nabla f(\mathbf{x}^*)+A(\mathbf{x}^*)\lambda^* = 0$. These two possibilities are illustrated in Question 4 in section 3.2.4.

Equation (3.2.3) is equivalent to the feasibility conditions $c_i(\mathbf{x}) = 0$ as

$$\frac{\partial}{\partial \lambda_i}\mathcal{L}(\mathbf{x},\lambda) = \frac{\partial}{\partial \lambda_i}\left(f(\mathbf{x}) + \sum_{i=1}^{m}\lambda_i c_i(\mathbf{x})\right) = c_i(\mathbf{x}).$$

Let $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ be defined by $\mathbf{c}(\mathbf{x}) = [\ c_1(\mathbf{x}) \quad \cdots \quad c_m(\mathbf{x})\ ]^T$, so

$$\mathcal{L}(\mathbf{x},\lambda) = f(\mathbf{x}) + \lambda^T\mathbf{c}(\mathbf{x})$$

Then

$$\begin{aligned}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\lambda) &= \nabla f(\mathbf{x}) + A(\mathbf{x})\lambda\\ \nabla_{\lambda}\mathcal{L}(\mathbf{x},\lambda) &= \mathbf{c}(\mathbf{x}).\end{aligned}$$

Thus (3.2.2) can be written as

$$\nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\lambda^* = 0. \tag{3.2.4}$$

Any regular local minimizer must satisfy both (3.2.2) and (3.2.3). However points which satisfy (3.2.2) and (3.2.3) may correspond to either a local minimizer a local maximizer or a constrained saddle point.

Definition 3.2.5 *Any point* $\mathbf{x}^*$ *for which there exists Lagrange multipliers* $\lambda^* \in \mathbb{R}^m$ *such that*

$$\begin{aligned}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\lambda^*) &= 0\\ \nabla_{\lambda}\mathcal{L}(\mathbf{x}^*,\lambda^*) &= 0\end{aligned}$$

is a constrained stationary point *of the equality constrained problem (3.2.1).*

A *constrained saddle point* is a constrained stationary point $\mathbf{x}^*$ such that for any $\delta > 0$ there exist $\mathbf{x},\mathbf{y} \in \Omega$ with $\|\mathbf{x}-\mathbf{x}^*\| \le \delta$, $\|\mathbf{y}-\mathbf{x}^*\| \le \delta$ and $f(\mathbf{x}) > f(\mathbf{x}^*)$ and $f(\mathbf{y}) < f(\mathbf{x}^*)$. That is in any feasible neighbourhood of $\mathbf{x}^*$ there are points with function values greater than $f(\mathbf{x}^*)$ and points with function value less that $f(\mathbf{x}^*)$.

The classical method for solving equality constrained optimization problems is the *method of Lagrange multipliers*. This consists of

1. Find all solutions $\mathbf{x}^* \in \mathbb{R}^n$ and $\lambda^* \in \mathbb{R}^m$ to the system of $n+m$ (nonlinear) equations

$$\nabla_x\mathcal{L}(\mathbf{x},\lambda) = \nabla f(\mathbf{x}) + A(\mathbf{x})\lambda^* = 0 \tag{3.2.5}$$

$$\nabla_\lambda cl\ (\mathbf{x},\lambda) = \mathbf{c}(\mathbf{x}) = 0 \tag{3.2.6}$$

2. Identify the solutions to (3.2.5) and (3.2.6) as either local minimizers, local maximizers or saddle points.

The feasible region $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{c}(\mathbf{x}) = 0\}$ may be unbounded, in which case there is no guarantee that global extrema exist. If the feasible region is not bounded then the behaviour of f as $\|\mathbf{x}\| \to \infty$ and $\mathbf{x} \in \Omega$ must be considered.

Example 3.2.6 *Example 3.2.1 has* $\mathcal{E} = \{1\}$,

$$f(\mathbf{x}) = x_1^2 + 6x_1x_2 + x_2^2 - 12x_1 - 4x_2 \qquad c_1(\mathbf{x}) = x_1 - x_2 + 1.$$

1. *Find the* unconstrained *stationary point of* $f(\mathbf{x})$. *Is it a constrained stationary point?*
2. *Show that the point* $\overline{\mathbf{x}} = [\ 0 \quad 1\ ]^T$ *is feasible but is* not *a constrained stationary point.*
3. *Find all the constrained stationary points.*

1. An unconstrained stationary point of $f(\mathbf{x})$ satisfies

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 + 6x_2 - 12 \\ 6x_1 + 2x_2 - 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Longrightarrow \hat{\mathbf{x}} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$

As this point has $c_1(\hat{\mathbf{x}}) = -1 \neq 0$ so $\hat{\mathbf{x}}$ is not feasible, so cannot be a constrained stationary point. The Hessian of f is

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 6 \\ 6 & 2 \end{bmatrix}$$

which is indefinite ($\det \nabla^2 f(\mathbf{x}) = -32 < 0$), so $\hat{\mathbf{x}}$ is an unconstrained saddle point.

2. The point $\overline{\mathbf{x}} = [\,0 \quad 1\,]^T$ has $c_1(\overline{\mathbf{x}}) = 0$ so is feasible. The Lagrangian function is

$$\mathcal{L}(\mathbf{x}, \lambda) = x_1^2 + 6x_1x_2 + x_2^2 - 12x_1 - 4x_2 + \lambda_1(x_1 - x_2 + 1).$$

A constrained stationary point for this equality constrained problem must satisfy

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) + \lambda_1 \nabla c_1(\mathbf{x}) = \begin{bmatrix} 2x_1 + 6x_2 - 12 + \lambda_1 \\ 6x_1 + 2x_2 - 4 - \lambda_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.2.7}$$

At $\overline{\mathbf{x}}$ this gives

$$\begin{bmatrix} -6 + \lambda_1 \\ -2 - \lambda_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which cannot be satisfied for any λ_1, so $\overline{\mathbf{x}}$ is not a constrained stationary point.

3. A constrained stationary point must be feasible, i.e.

$$c_1(\mathbf{x}) = x_1 - x_2 + 1 = 0,$$

and satisfy (3.2.7). Solving these three equations gives

$$\mathbf{x}^* = \begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \end{bmatrix} \quad \text{and} \quad \lambda_1^* = 2$$

as the only constrained stationary point. From the plots in Figure 3.2.1 it is clear that this is a minimizer of the equality constrained problem. However second order conditions are needs to verify this.

Example 3.2.7 *Find all constrained stationary points for the problem*

$$\begin{array}{ll} \textit{Minimize} & -x_1x_2x_3 \\ \textit{Subject to} & x_1 + 2x_2 + 2x_3 = 72. \end{array}$$

Here $n = 3$, $f(\mathbf{x}) = -x_1x_2x_3$, $m = 1$, $\mathcal{E} = \{1\}$ and $c_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 - 72$. The Lagrangian function is

$$\mathcal{L}(\mathbf{x}, \lambda) = -x_1x_2x_3 + \lambda_1(x_1 + 2x_2 + 2x_3 - 72).$$

Thus (3.2.2) and (3.2.3) give

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{x_1} = -x_2x_3 + \lambda_1 = 0 \tag{3.2.8}$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{x_2} = -x_1x_3 + 2\lambda_1 = 0 \tag{3.2.9}$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{x_3} = -x_1x_2 + 2\lambda_1 = 0 \tag{3.2.10}$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{\lambda_1} = (x_1 + 2x_2 + 2x_3 - 72) = 0 \tag{3.2.11}$$

All the solutions to (3.2.8) to (3.2.11) are required. From (3.2.8) $\lambda_1 = x_2 x_3$. Substituting in (3.2.9) and (3.2.10) produces

$$x_3(-x_1 + 2x_2) = 0 \tag{3.2.12}$$

$$x_2(-x_1 + 2x_3) = 0 \tag{3.2.13}$$

Equation (3.2.12) implies either $x_3 = 0$ or $x_1 = 2x_2$, while equation (3.2.13) implies either $x_2 = 0$ or $x_1 = 2x_3$. Going through all the possible cases produces the constrained stationary points

$$\mathbf{x}^{(a)} = \begin{bmatrix} 72 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(b)} = \begin{bmatrix} 0 \\ 36 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(c)} = \begin{bmatrix} 0 \\ 0 \\ 36 \end{bmatrix} \quad \mathbf{x}^{(d)} = \begin{bmatrix} 24 \\ 12 \\ 12 \end{bmatrix}$$

$$\lambda_1^{(a)} = 0 \qquad \lambda_1^{(b)} = 0 \qquad \lambda_1^{(c)} = 0 \qquad \lambda_1^{(d)} = 144$$

$$f(\mathbf{x}^{(a)}) = 0 \qquad f(\mathbf{x}^{(b)}) = 0 \qquad f(\mathbf{x}^{(c)}) = 0 \qquad f(\mathbf{x}^{(d)}) = -3456$$

Points along the line

$$\mathbf{x}(\alpha) = \begin{bmatrix} -2\alpha \\ \alpha \\ 36 \end{bmatrix}$$

are always feasible, with $f(\mathbf{x}(\alpha) = 72\alpha^2$. Thus $f(\mathbf{x}(\alpha)) \to \infty$ as $\alpha \to \infty$, so the problem does not have a global maximum. Similarly points along the line

$$\mathbf{x}(\alpha) = \begin{bmatrix} 144 - 2\alpha \\ \alpha \\ -36 \end{bmatrix}$$

are always feasible, and $f(\mathbf{x}(\alpha) = 72\alpha(72 - \alpha)$. Thus $f(\mathbf{x}(\alpha)) \to -\infty$ as $\alpha \to \infty$, so the problem does not have a global minimum.

As global extrema do not exist for this problem, the nature of the constrained stationary points cannot be determined from first order derivative information.

Any set $\{\mathbf{a}\}$, $\mathbf{a} \neq 0$ consisting of a single non-zero vector, is linearly independent. For this example any feasible point has $A(\mathbf{x}) = [\,1 \quad 2 \quad 2\,]^T$ which is full rank, so any feasible point is a regular point.

3.2.2 Second order conditions

Second order information may help determine the nature of a constrained stationary point.

Proposition 3.2.8 (Second order necessary conditions) *Let $f \in C^2(\Omega)$. Let $\mathbf{x}^*$ be a local minimizer and a regular point of (3.2.1). Then there exist multipliers $\lambda^* \in \mathbb{R}^m$ such that*

$$\begin{aligned} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) &= \nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\lambda^* = 0, \\ \nabla_{\lambda} \mathcal{L}(\mathbf{x}^*, \lambda^*) &= \mathbf{c}(\mathbf{x}^*) = 0. \end{aligned}$$

and

$$\mathbf{d}^T \nabla_x^2 \mathcal{L}(\mathbf{x}^*, \lambda^*)\mathbf{d} \geq 0 \qquad \textit{for all } \mathbf{d} \textit{ such that } \mathbf{d}^T A(\mathbf{x}^*) = 0. \tag{3.2.14}$$

Proposition 3.2.9 (Second order sufficient conditions) *Let $f \in C^2(\Omega)$. Let $\mathbf{x}^*$ be a constrained stationary point, so there exist multipliers $\lambda^* \in \mathbb{R}^m$ such that*

$$\begin{aligned} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) &= \nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\lambda^* = 0, \\ \nabla_{\lambda} \mathcal{L}(\mathbf{x}^*, \lambda^*) &= \mathbf{c}(\mathbf{x}^*) = 0. \end{aligned}$$

If

$$\mathbf{d}^T \nabla_x^2 \mathcal{L}(\mathbf{x}^*, \lambda^*)\mathbf{d} > 0 \qquad \textit{for all } \mathbf{d} \neq 0 \textit{ such that } \mathbf{d}^T A(\mathbf{x}^*) = 0 \tag{3.2.15}$$

then $\mathbf{x}^*$ is a strict local minimizer of the equality constrained problem.

The proofs of both Proposition 3.2.8 and 3.2.9 cna be found in Fletcher [Fle87].

The Hessian of the Lagrangian function at $\mathbf{x}^*, \lambda^*$ is

$$W^* = \nabla_x^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) = \nabla^2 f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla^2 c_i(\mathbf{x}^*). \tag{3.2.16}$$

Geometrically the restriction that $\mathbf{d}^T A(\mathbf{x}^*) = 0$ means that the direction $\mathbf{d}$ is tangential to the constraint surface at $\mathbf{x}^*$. If $\operatorname{rank} A(\mathbf{x}^*) = n$, then the only solution to $\mathbf{d}^T A(\mathbf{x}^*) = 0$ is $\mathbf{d} = 0$. In this case the second order conditions do not give any information about the nature of the constrained stationary point, as the n equality constraints imply the feasible region consists of a single point. When $\operatorname{rank} A(\mathbf{x}^*) < n$ then (3.2.15) means that the curvature of the Lagrangian function in the space tangent to the constraints is positive.

When $\operatorname{rank} A(\mathbf{x}^*) < n$ a basis for the null space of $A(\mathbf{x}^*)^T$ can be used to efficiently represent the tangent space to the constraint surface. Let $\operatorname{rank} A(\mathbf{x}^*) = t^*$ and let the columns of $Z^* \in \mathbb{R}^{n \times n-t^*}$ form a basis for the null space of $A(\mathbf{x}^*)^T$, so

$$A(\mathbf{x}^*)^T Z^* = 0 \quad \text{and} \quad \operatorname{rank} Z^* = n - t^*. \tag{3.2.17}$$

Numerically it is desirable to use an orthonormal basis Z^*, so

$$Z^{*T} Z^* = I_{n-t^*}.$$

An orthonormal basis can be obtained from the QR factorization of $A(\mathbf{x}^*)$ (see Appendix A). For any vector $\mathbf{d}$ such that $\mathbf{d}^T A(\mathbf{x}^*) = 0$ there exists a $\mathbf{v} \in \mathbb{R}^{n-t^*}$ such that $\mathbf{d} = Z^* \mathbf{v}$. Equation (3.2.15) is equivalent to

$$\mathbf{v}^T Z^{*T} \nabla_x^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^* \mathbf{v} > 0 \quad \forall \mathbf{v} \in \mathbb{R}^{n-t^*}, \mathbf{v} \neq 0,$$

so that the *Reduced Hessian*

$$W_R^* = Z^{*T} \nabla_x^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^* \tag{3.2.18}$$

is positive definite. (3.2.18) is the Hessian of the Lagrangian function evaluated at $\mathbf{x}^*, \lambda^*$, projected into the space tangent to the constraints at $\mathbf{x}^*$.

Let $\mathbf{x}^*$ be a constrained stationary point of the equality constrained problem (3.2.1), so (3.2.5) and (3.2.6) are satisfied. Then

1. $Z^{*T} \nabla^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^*$ positive definite $\Longrightarrow$ $\mathbf{x}^*$ is a strict local minimizer of f on Ω.
2. $Z^{*T} \nabla^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^*$ negative definite $\Longrightarrow$ $\mathbf{x}^*$ is a strict local maximizer of f on Ω.
3. $Z^{*T} \nabla^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^*$ indefinite $\Longrightarrow$ $\mathbf{x}^*$ is a constrained saddle point.

Example 3.2.10 *Example 3.2.1 has a constrained stationary point* $\mathbf{x}^* = [\ \frac{1}{2} \quad \frac{3}{2}\ ]^T$ *(see Example 3.2.6). Use the second order conditions to determine, if possible, the nature of* $\mathbf{x}^*$.

The constraint gradient at $\mathbf{x}^*$ is

$$\nabla c_1(\mathbf{x}^*) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Hence a full rank matrix $Z \in \mathbb{R}^{2 \times 1}$ that is orthogonal to the constraint gradients ($Z^T \nabla c_1(\mathbf{x}^*) = 0$) is

$$Z = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The Hessian of the Lagrangian function at $\mathbf{x}^*, \lambda^*$ is

$$\nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) = \nabla^2 f(\mathbf{x}^*) + \lambda_1^* \nabla^2 c_1(\mathbf{x}^*) = \begin{bmatrix} 2 & 6 \\ 6 & 2 \end{bmatrix},$$

so the reduced Hessian is

$$Z^T\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\lambda^*)Z = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 6 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 16 > 0.$$

As the reduced Hessian $Z^T\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\lambda^*)Z$ is positive definite the constrained stationary point $\mathbf{x}^*$ is a strict local minimizer with $f(\mathbf{x}^*) = -5$. As the constraint in this example is affine, $\nabla^2 c_1(\mathbf{x}) = 0$, so the curvature of the Lagrangian comes only from the curvature of the objective function.

Example 3.2.11 *Example 3.2.2 has* $\mathcal{E} = \{1\}$,

$$f(\mathbf{x}) = -x_1 + x_2 + 5 \quad \textit{and} \quad c_1(\mathbf{x}) = -x_1^2 + x_2$$

Show that $\mathbf{x}^* = [\ \frac{1}{2} \quad \frac{1}{4}\ ]^T$ *is the only constrained stationary point and use the second order conditions to determine its nature.*

A constrained stationary point must satisfy

$$\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\lambda) = \nabla f(\mathbf{x}) + \sum_{i\in\mathcal{E}} \lambda_i \nabla c_i(\mathbf{x}) = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1 \begin{bmatrix} -2x_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

and

$$\nabla_\lambda \mathcal{L}(\mathbf{x},\lambda) = \mathbf{c}(\mathbf{x}) = -x_1^2 + x_2 = 0.$$

Solving these three equations shows that the only solution is

$$\mathbf{x}^* = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{4} \end{bmatrix}, \quad \lambda_1^* = -1.$$

The constraint gradient at $\mathbf{x}^*$ is

$$\nabla c_1(\mathbf{x}^*) = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \Longrightarrow Z = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The Hessian of the Lagrangian function is

$$\nabla^2 cl\ (\mathbf{x}^*,\lambda^*) = \nabla^2 f(\mathbf{x}^*) + \lambda_1 \nabla^2 c_1(\mathbf{x}^*) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + (-1) \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix},$$

so the reduced Hessian is

$$Z^T\nabla^2\mathcal{L}(\mathbf{x}^*,\lambda^*)Z = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 2 > 0.$$

Hence $\mathbf{x}^*$ is a strict local minimizer of $f(\mathbf{x})$ over the region $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) = 0\}$. The minimum value is $f(\mathbf{x}^*) = 4.75$. In this example the objective function is affine, so the curvature of the Lagrangian function comes entirely from the curvature of the constraint function.

Example 3.2.12 *For Example 3.2.7 the constrained stationary points are*

$$\mathbf{x}^{(a)} = \begin{bmatrix} 72 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(b)} = \begin{bmatrix} 0 \\ 36 \\ 0 \end{bmatrix} \quad \mathbf{x}^{(c)} = \begin{bmatrix} 0 \\ 0 \\ 36 \end{bmatrix} \quad \mathbf{x}^{(d)} = \begin{bmatrix} 24 \\ 12 \\ 12 \end{bmatrix}$$

$$\lambda_1^{(a)} = 0 \qquad \lambda_1^{(b)} = 0 \qquad \lambda_1^{(c)} = 0 \qquad \lambda_1^{(d)} = 144$$

$$f(\mathbf{x}^{(a)}) = 0 \qquad f(\mathbf{x}^{(b)}) = 0 \qquad f(\mathbf{x}^{(c)}) = 0 \qquad f(\mathbf{x}^{(d)}) = -3456$$

The gradients of the objective function and the constraint are

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -x_2x_3 \\ -x_1x_3 \\ -x_1x_2 \end{bmatrix}, \qquad \nabla c_1(\mathbf{x}) = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = A(\mathbf{x}).$$

As $\nabla^2 c_1(\mathbf{x}) = 0$ *the Hessian of the Lagrangian function is*

$$\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x},\lambda) = \nabla^2 f(\mathbf{x}) + \lambda_1 \nabla^2 c_1(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 0 & -x_3 & -x_2 \\ -x_3 & 0 & -x_1 \\ -x_2 & -x_1 & 0 \end{bmatrix}.$$

Here A *is independent of* $\mathbf{x}$ *and has full rank, so* $t^*1, n - t^* = 2$ *and* $Z \in \mathbb{R}^{3\times 2}$. *One possible choice is*

$$Z = \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}.$$

The reduced Hessian of the Lagrangian at $\mathbf{x}^{(a)}, \lambda^{(a)}$ *is*

$$Z^T\nabla^2_x\mathcal{L}(\mathbf{x}^{(a)},\lambda^{(a)})Z = \begin{bmatrix} 0 & 1 & -1 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -72 \\ 0 & -72 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 144 & -72 \\ -72 & 0 \end{bmatrix}.$$

The determinant of this matrix is $-72^2 < 0$, *so the reduced Hessian is indefinite. Thus* $\mathbf{x}^{(a)}, \lambda^{(a)}$ *is a constrained saddle point.*

Similarly

$$Z^T\nabla^2_x\mathcal{L}(\mathbf{x}^{(b)},\lambda^{(b)})Z = \begin{bmatrix} 0 & 72 \\ 72 & 0 \end{bmatrix}$$

is indefinite, so $\mathbf{x}^{(b)}, \lambda^{(b)}$ *is a constrained saddle point. Also*

$$Z^T\nabla^2_x\mathcal{L}(\mathbf{x}^{(c)},\lambda^{(c)})Z = \begin{bmatrix} 0 & -72 \\ -72 & -144 \end{bmatrix}$$

is indefinite, so $\mathbf{x}^{(c)}, \lambda^{(c)}$ *is a constrained saddle point. However*

$$Z^T\nabla^2_x\mathcal{L}(\mathbf{x}^{(d)},\lambda^{(d)})Z = \begin{bmatrix} 0 & 1 & -1 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -12 & -12 \\ -12 & 0 & -12 \\ -12 & -12 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & -1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 48 & -24 \\ -24 & 48 \end{bmatrix}$$

is positive definite as the product of its eigenvalue (the determinant) is 1728 *and the sum of its eigenvalues (the trace) is* 96. *Hence* $\mathbf{x}^{(d)}, \lambda^{(d)}$ *is a strict local minimizer.*

In Example 3.2.12 the Hessian of the objective function at $\mathbf{x}^{(d)}$ is

$$\nabla^2 f(\mathbf{x}^{(d)}) = \begin{bmatrix} 0 & -12 & -12 \\ -12 & 0 & -24 \\ -12 & -24 & 0 \end{bmatrix},$$

which is indefinite as it has eigenvalues $24, 8.7846, -32.7846$. However the reduced Hessian is positive definite with eigenvalues 24 and 72. If a symmetric matrix $H \in \mathbb{R}^{n\times n}$ is positive definite and $Z \in \mathbb{R}^{n\times t}$ has full rank $0 < t \le n$ then the reduced matrix $Z^THZ \in \mathbb{R}^{t\times t}$ is symmetric and positive definite. Thus if the Hessian of the Lagrangian function is positive definite the reduced Hessian is automatically positive definite.

Example 3.2.13 *Let $\mathbf{g} \in \mathbb{R}^n, \mathbf{g} \neq 0$. Find the extrema of $\mathbf{d}^T\mathbf{g}$ over the set $\Omega = \{\mathbf{d} \in \mathbb{R}^n : \mathbf{d}^T\mathbf{d} = 1\}$, and determined which are local or global minimizers or maximizers. The Lagrangian function is*

$$\mathcal{L}(\mathbf{d}, \lambda) = \mathbf{d}^T\mathbf{g} + \lambda_1(\mathbf{d}^T\mathbf{d} - 1).$$

The necessary conditions for a local extrema are

$$\nabla_{\mathbf{d}}\mathcal{L}(\mathbf{d}, \lambda) = \mathbf{g} + 2\lambda_1\mathbf{d} = 0 \tag{3.2.19}$$
$$\nabla_{\lambda}\mathcal{L}(\mathbf{d}, \lambda) = \mathbf{d}^T\mathbf{d} - 1 = 0 \tag{3.2.20}$$

As $\mathbf{g} \neq 0$, equation (3.2.20) implies $\lambda_1 \neq 0$, and hence

$$\mathbf{d} = -\frac{1}{2\lambda_1}\mathbf{g}.$$

Substituting in equation (3.2.20) gives

$$\mathbf{d}^T\mathbf{d} = \frac{1}{4\lambda_1^2}\mathbf{g}^T\mathbf{g} = 1 \quad \Longrightarrow \quad \lambda_1 = \pm\frac{\|\mathbf{g}\|}{2}.$$

Thus the constrained stationary points are

$$\bar{d} = \frac{\mathbf{g}}{\|\mathbf{g}\|}, \bar{\lambda}_1 = -\frac{\|\mathbf{g}\|}{2} \quad \textit{and} \quad \hat{d} = -\frac{\mathbf{g}}{\|\mathbf{g}\|}, \hat{\lambda}_1 = \frac{\|\mathbf{g}\|}{2}$$

The Hessian of the Lagrangian function is

$$\nabla^2_{\mathbf{d}}\mathcal{L}(\mathbf{d}, \lambda) = 2\lambda_1 I.$$

At $\bar{\mathbf{d}}$ the multiplier $\bar{\lambda}_1 < 0$, so the Hessian of the Lagrangian (and hence any projection of it) is negative definite. Thus $\bar{\mathbf{d}}, \bar{\lambda}_1$ corresponds to strict local maximizer. At $\hat{\mathbf{d}}$ the multiplier $\bar{\lambda}_1 > 0$ so the Hessian of the Lagrangian (and hence any projection of it) is positive definite. Hence $\hat{\mathbf{d}}, \hat{\lambda}_1$ corresponds to s strict local minimizer.

As the objective function $f(\mathbf{d}) = \mathbf{d}^T\mathbf{g}$ is continuous, and the feasible region $\Omega = \{\mathbf{d} \in \mathbb{R}^n : \|\mathbf{d}\| = 1\}$ is closed and bounded, global extrema exist for this problem. As the only *constrained stationary points are $\bar{\mathbf{d}}, \bar{\lambda}$ and $\hat{\mathbf{d}}, \hat{\lambda}$, it follows that $\bar{\mathbf{d}}, \bar{\lambda}_1$ is the global maximizer, and $\hat{\mathbf{d}}, \hat{\lambda}_1$ is the global minimizer.*

Let $\mathbf{g} = \nabla f(\mathbf{x})$ in Example 3.2.13. Then Example 3.2.13 shows that the direction (with length 1) having the greatest slope at $\mathbf{x}$ is the direction $\mathbf{d} = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$, and the direction having the least slope at $\mathbf{x}$ is the direction $\mathbf{d} = -\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$. Hence $\nabla f(\mathbf{x})$ is referred to as the *steepest ascent* direction, and $-\nabla f(\mathbf{x})$ is referred to as the *steepest descent* direction.

3.2.3 Sensitivity analysis

Sensitivity analysis looks at how the solution value and solution point change when the problem data changes. Suppose the jth equality constraint changes from

$$c_j(\mathbf{x}) = 0 \quad \text{to} \quad c_j(\mathbf{x}) + \epsilon = 0.$$

Let $\mathbf{x}^*(\epsilon), \lambda^*(\epsilon)$ denote a solution to the first order necessary conditions with the perturbed constraint, so

$$\nabla f(\mathbf{x}^*(\epsilon)) + \sum_{i=1}^{m} \lambda_i^*(\epsilon)\nabla c_i(\mathbf{x}^*(\epsilon)) = 0$$
$$\mathbf{c}(\mathbf{x}^*(\epsilon)) + \epsilon\mathbf{e}_j = 0,$$

where $\mathbf{e}_j \in \mathbb{R}^m$ is the jth unit vector. The original solution corresponds to $\epsilon = 0$, i.e. $\mathbf{x}^* = \mathbf{x}^*(0)$.

The change in the objective value from perturbing the jth equality constraint by ϵ is

$$\begin{aligned}\Delta f(\mathbf{x}^*) = f(\mathbf{x}^*(\epsilon)) - f(\mathbf{x}^*) &= \mathcal{L}(\mathbf{x}^*(\epsilon), \lambda^*(\epsilon)) - \mathcal{L}(\mathbf{x}^*, \lambda^*) \\ &\approx \frac{\partial \mathcal{L}(\mathbf{x}^*, \lambda^*)}{\partial c_j} \Delta c_j = \lambda_j^* \epsilon.\end{aligned}$$

Thus λ_j^* gives a first order estimate of the rate of change of the optimal objective value with respect to changes in the jth constraint. As

$$c_j(\mathbf{x}) + \epsilon = 0 \iff c_j(\mathbf{x}) = -\epsilon$$

λ_j^* gives an estimate of the rate of change of the optimal objective value with respect to the negative of changes to the right-hand-side of the jth constraint.

Example 3.2.14 *Examples 3.2.7 and 3.2.12 considered the problem*

$$\begin{array}{ll} Minimize & -x_1 x_2 x_3 \\ Subject\ to & x_1 + 2x_2 + 2x_3 = 72. \end{array}$$

which has a strict local minimizer with

$$\mathbf{x}^* = \begin{bmatrix} 24 \\ 12 \\ 12 \end{bmatrix}, \quad \lambda_1^* = 144. \quad f(\mathbf{x}^*) = -3456.$$

Suppose the constant 72 in the constraint changes to 75. This corresponds to a perturbation $\epsilon = -3$ as $c_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 - 72 = 0$ becomes $c_1(\mathbf{x}) + \epsilon = 0$. A first order estimate of the change in the objective value is

$$\Delta f^* = \lambda_1^* \epsilon = 144 \times -3 = -432.$$

To check the accuracy of this estimate resolve the problem with the constraint

$$\hat{c}_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 - 72 + \epsilon = 0.$$

The first order necessary conditions give

$$\nabla f(\mathbf{x}(\epsilon)) + A(\mathbf{x}(\epsilon))\lambda(\epsilon) = 0 \Longrightarrow \begin{bmatrix} -x_2 x_3 \\ -x_1 x_3 \\ -x_1 x_2 \end{bmatrix} + \lambda_1 \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = 0 \Longrightarrow \lambda_1 = x_2 x_3 \text{ and } x_1 = 2x_2 = 2x_3.$$

Substituting in the constraint

$$\hat{c}_1(\mathbf{x}(\epsilon)) = 0 \Longrightarrow 3x_1 - 72 + \epsilon = 0$$

Hence

$$\mathbf{x}^*(\epsilon) = \begin{bmatrix} 24 - \frac{\epsilon}{3} \\ 12 - \frac{\epsilon}{6} \\ 12 - \frac{\epsilon}{6} \end{bmatrix}, \ \lambda^*(\epsilon) = \left(12 - \frac{\epsilon}{6}\right)^2 \text{ and } f(\mathbf{x}^*(\epsilon)) = -(24 - \frac{\epsilon}{3})(12 - \frac{\epsilon}{6})^2.$$

The function value $f(\mathbf{x}^*(\epsilon))$ at the minimum of the perturbed problem and the linear approximation $f(\mathbf{x}^*) + \lambda_1^* \epsilon$ based on the Lagrange multiplier are sketched in Figure 3.2.3. The actual change in the function value is

$$f(\mathbf{x}^*(\epsilon)) - f(\mathbf{x}^*) = -\left(24 - \frac{\epsilon}{3}\right)\left(12 - \frac{\epsilon}{6}\right)^2 + 3456 = 144\epsilon - 2\epsilon^2 + \frac{1}{108}\epsilon^3.$$

A first order estimate of this change is

$$\left.\frac{d\ (f(\mathbf{x}^*(\epsilon)) - f(\mathbf{x}^*))}{d\epsilon}\right|_0 \epsilon = 144 - 4\epsilon + \left.\frac{1}{36}\epsilon^2\right|_0 \epsilon = 144\epsilon.$$

This provides a good estimate for small values of ϵ. For example $\epsilon = -0.1$ gives a predicted change of -14.4, compared to the actual change in the objective of -14.420012. However for $\epsilon = -3$ the predicted change in the objective function of -432 does not compare so well with the actual change of -450.12.

The objective function is the negative of the volume of a rectangular parcel. The constraint is the maximum value for the length plus the girth. Increasing this from a maximum of 72 inches to 75 inches should allow the volume of the largest parcel to increase. The predicted increase is 432 cubic inches from a volume of $3,456$ cubic inches. The actual increase would be 450.12 cubic inches.

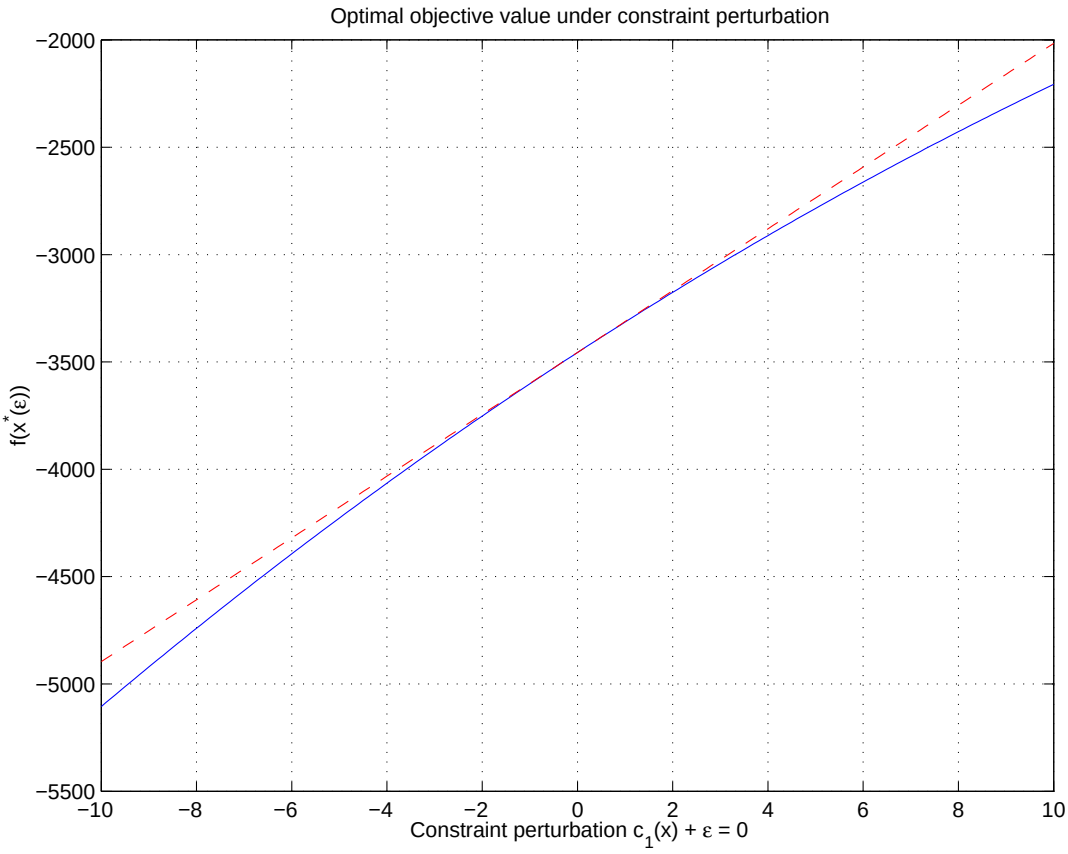


Figure 3.2.3: Optimal function value $f(\mathbf{x}^*(\epsilon))$ and linear approximation $f(\mathbf{x}^*) + \lambda_j^* \epsilon$

3.2.4 Exercises

1. Find the global extrema of $f(\mathbf{x}) = x_1 x_2^2$ subject to $x_1^2 + x_2^2 = 1$.

   Ans: Global minimizers $[\ -1/\sqrt{3} \quad \pm\sqrt{2/3}\ ]^T$, global maximizers $[\ 1/\sqrt{3} \quad \pm\sqrt{2/3}\ ]^T$.

2. Consider the problem:

$$\begin{array}{ll} \textit{Minimize} & f(\mathbf{x}) = x_1^2 + x_2^2 + 3x_2 \\ \textit{Subject to} & c_1(\mathbf{x}) = x_1^2 + (x_2+1)^2 - 1 = 0. \end{array}$$

   (a) Show that $\bar{\mathbf{x}} = [\ 0 \quad 0\ ]^T$ and $\mathbf{x}^* = [\ 0 \quad -2\ ]^T$ are regular points and constrained stationary points.

   (b) Show that $\nabla^2 f(\bar{\mathbf{x}})$ is positive definite, yet $\bar{\mathbf{x}}$ is not a local minimizer.

   Hint: Show that the feasible region Ω can be parametrized as

$$\left\{ \mathbf{x} \in \mathbb{R}^2 : x_1 = \cos(\theta), x_2 = \sin(\theta) - 1, \quad \theta \in [0, 2\pi) \right\},$$

   and find f as a function of θ.

   (c) Show that $\mathbf{x}^*$ is a strict local minimizer.

3. Let $f(\mathbf{x}) = x_1^2 + x_2^2 + 3x_2$ and $c_1(\mathbf{x}) = x_1^2 + (x_2+1)^2 - 1 + \epsilon = 0$.

   (a) Find the local minimizer $\mathbf{x}^*(\epsilon)$ such that $\mathbf{x}^*(0) = [\ 0 \quad -2\ ]^T$, and calculate the function value $f^*(\epsilon) \equiv f(\mathbf{x}^*(\epsilon))$ and the Lagrange multiplier $\lambda_1^*(\epsilon)$.

(b) Show that

$$\frac{df^*(\epsilon)}{d\epsilon} = \lambda_1^*(\epsilon).$$

(c) Use $\Delta f^* \approx \lambda_1^* \epsilon$ to estimate the change in the minimum value when $x_1^2 + (x_2+1)^2 = 2$ and when $x_1^2 + (x_2+1)^2 = 1.1$. Compare this with the actual changes $f^*(\epsilon) - f^*(0)$, and explain any differences.

4. Consider minimizing $f(\mathbf{x}) = x_1$ subject to $x_2 = x_1^2$ and $x_2 = 0$.

(a) Show that $\mathbf{x}^* = [\ 0 \quad 0\ ]^T$ is the only feasible point, and hence the minimizer.

(b) Show that $\mathbf{x}^*$ is NOT a regular point.

(c) Do multipliers λ^* satisfying

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{E}} \lambda_i \nabla c_i(\mathbf{x}^*) = 0 \tag{3.2.21}$$

exist?

(d) What happens if the objective function is changed to $f(\mathbf{x}) = x_2$?

5. Find a matrix Z whose columns form a basis for the tangent space to the constraints

$$\begin{aligned} c_1(\mathbf{x}) &\equiv x_1^2 + x_2^2 + x_3^2 + x_4^2 - 4 = 0 \\ c_2(\mathbf{x}) &\equiv x_2^3 + 5x_4 - 6 = 0, \end{aligned}$$

at the point $\mathbf{x}^* = [\ 1 \quad 1 \quad 1 \quad 1\ ]^T$.

(a) Find Z by solving the system of linear equations that the elements of Z must satsify.

(b) Find Z from a QR factorization (use either MATLAB or `Maple`) of the matrix $A = \left[\ \nabla c_1(\mathbf{x}^*) \quad \nabla c_2(\mathbf{x}^*)\ \right]$.

3.3 Inequality Constraints

When inequality constraints are included the standard problem is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & f(\mathbf{x}) \\ Subject\ to & c_i(\mathbf{x}) = 0 \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}) \le 0 \quad i \in \mathcal{I}. \end{array} \tag{3.3.1}$$

The feasible region is

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0\ i \in \mathcal{E}, c_i(\mathbf{x}) \le 0\ i \in \mathcal{I}\}.$$

Without loss of generality it is assumed that $\mathcal{E} = \{1, \ldots, m_{\mathcal{E}}\}$ and $\mathcal{I} = \{m_{\mathcal{E}}+1, \ldots, m\}$, where $m = |\mathcal{E}| + |\mathcal{I}|$ is the total number of constraints.

Associated with each constraint is a Lagrange multiplier λ_i for $i = 1, \ldots, m$. The Lagrangian function is

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i c_i(\mathbf{x}) = f(\mathbf{x}) + \lambda^T \mathbf{c}(\mathbf{x}) \tag{3.3.2}$$

where $\lambda \in \mathbb{R}^m$ and $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$.

Constraints which are satisfied as equalities at a point $\mathbf{x}$ are *active* at $\mathbf{x}$. Only the active constraints are binding at a solution.

Definition 3.3.1 *The* set of active constraints *at a point* $\mathbf{x}$ *is*

$$\mathcal{A}(\mathbf{x}) = \{i \in 1, \ldots, m : c_i(\mathbf{x}) = 0\}.$$

At any feasible point the equality constraints (3.3.4c) must be satisfied, so

$$\mathcal{A}(\mathbf{x}) = \mathcal{E} \cup \{\, i \in \mathcal{I} : c_i(\mathbf{x}) = 0 \,\} = \{\, 1, \ldots, m_{\mathcal{E}} \,\} \cup \{\, i \in \mathcal{I} : c_i(\mathbf{x}) = 0 \,\}.$$

Definition 3.3.2 *A feasible point* $\mathbf{x}$ *is a* regular point *if the gradients of the active constraints,* $\nabla c_i(\mathbf{x})$ $i \in \mathcal{A}(\mathbf{x})$ *are linearly independent.*

Example 3.3.3 (IQEX1) *Consider the problem with*

$$f(\mathbf{x}) \equiv -x_1^2 - (x_2-1)^2(x_2-3)^2 - \frac{x_2}{2} \tag{3.3.3a}$$

$$c_1(\mathbf{x}) \equiv x_1^2 - x_2 \tag{3.3.3b}$$

$$c_2(\mathbf{x}) \equiv -x_1 + x_2 - 2 \tag{3.3.3c}$$

$$c_3(\mathbf{x}) \equiv x_1 + x_2 - 4. \tag{3.3.3d}$$

For this problem $n = 2$, $m = 3$, $\mathcal{E} = \emptyset$ *and* $\mathcal{I} = \{1, 2, 3\}$. *Figure 3.3.1 contains a contour plot with the constraints drawn as dashed lines and a surface plot of the objective over the feasible region. Consider the*

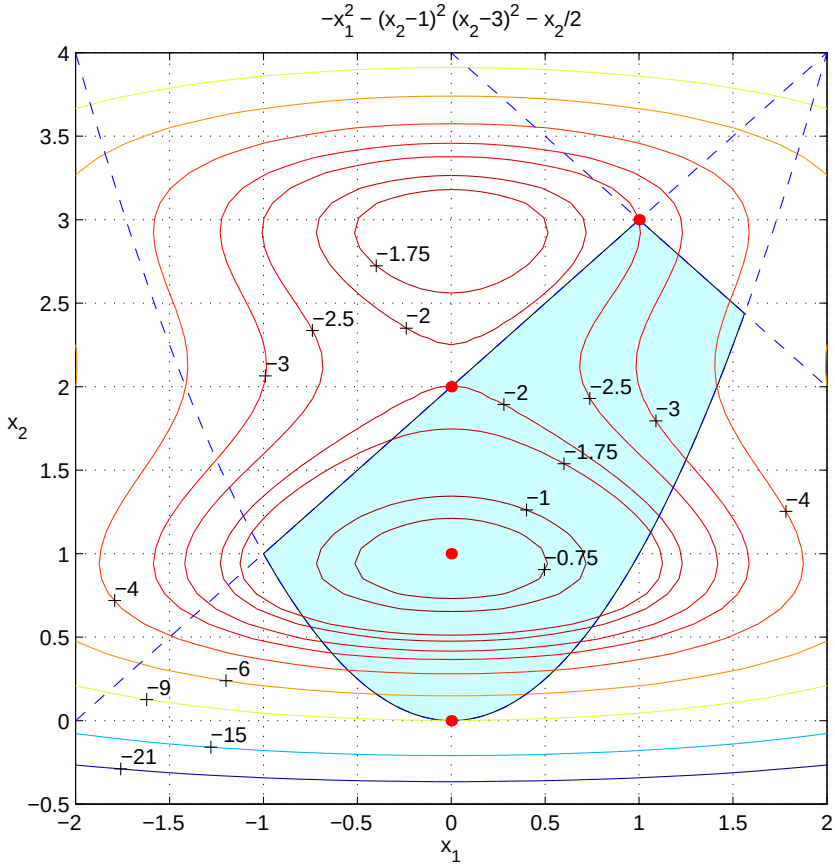


Figure 3.3.1: Feasible region Ω and contour plot of f

points

$$\tilde{\mathbf{x}} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \quad \overline{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \textit{and} \quad \hat{\mathbf{x}} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

Evaluating the constraint functions at these points gives

$$\mathbf{c}(\tilde{\mathbf{x}}) = \begin{bmatrix} -2 \\ 0 \\ -2 \end{bmatrix}, \quad \mathbf{c}(\overline{\mathbf{x}}) = \begin{bmatrix} 0 \\ -2 \\ -4 \end{bmatrix} \quad \textit{and} \quad \mathbf{c}(\hat{\mathbf{x}}) = \begin{bmatrix} -2 \\ 0 \\ 0 \end{bmatrix}.$$

As there are no equality constraints and $\mathbf{c}(\tilde{\mathbf{x}}) \leq 0, \mathbf{c}(\overline{\mathbf{x}}) \leq 0$ *and* $\mathbf{c}(\hat{\mathbf{x}}) \leq 0$*, all three points* $\tilde{\mathbf{x}}, \overline{\mathbf{x}}$ *and* $\hat{\mathbf{x}}$ *are feasible. The sets of active constraints are* $\mathcal{A}(\tilde{\mathbf{x}}) = \{2\}$*,* $\mathcal{A}(\overline{\mathbf{x}}) = \{1\}$ *and* $\mathcal{A}(\hat{\mathbf{x}}) = \{2, 3\}$*. The constraints* $c_2(\mathbf{x})$ *and* $c_3(\mathbf{x})$ *which are not active at* $\overline{\mathbf{x}}$ *can be changed a little without affecting the nature of* $\overline{\mathbf{x}}$*. Similarly the constraint* $c_1(\mathbf{x})$ *can be changed slightly without affecting* $\hat{\mathbf{x}}$*. It is only the constraints that are active at* $\mathbf{x}$ *that help determine the nature of the point* $\mathbf{x}$*.*

3.3.1 First order conditions

Theorem 3.3.4 (Karush-Kuhn-Tucker (KKT) conditions) *Let* f *and* $c_i, i = 1, \ldots, m$ *be once continuously differentiable on* Ω*. If* $\mathbf{x}^*$ *is a local minimizer of (3.3.1) and a regular point then there exist multipliers* $\lambda_i^*, i = 1, \ldots, m$ *such that*

$$\nabla_x \mathcal{L}(\mathbf{x}^*, \lambda^*) = \nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0 \tag{3.3.4a}$$

$$c_i(\mathbf{x}^*) = 0 \quad i = 1, \ldots, m_{\mathcal{E}} \tag{3.3.4b}$$

$$c_i(\mathbf{x}^*) \leq 0 \quad i = m_{\mathcal{E}} + 1, \ldots, m \tag{3.3.4c}$$

$$\lambda_i \geq 0 \quad i = m_{\mathcal{E}} + 1, \ldots, m \tag{3.3.4d}$$

$$\lambda_i^* c_i(\mathbf{x}^*) = 0 \quad i = 1, \ldots, m \tag{3.3.4e}$$

The KKT conditions (3.3.4) must be satisfied at any local minimizer which is a regular point. Equations (3.3.4c) and (3.3.4d) represent the feasibility of $\mathbf{x}^*$. The non-negativity conditions (3.3.4d) on the multipliers only apply to the inequality constraints; there are no sign restrictions on the multipliers corresponding to equality constraints. Equations (3.3.4e) are the *complementary conditions.* If an inequality constraint is feasible, but not satisfied as an equality, then $c_i(\mathbf{x}^*) > 0$. The complementarity condition $\lambda_i^* c_i(\mathbf{x}^*) = 0$ then implies that the corresponding multiplier $\lambda_i^* = 0$.

As inactive constraints must have zero multipliers the KKT conditions (3.3.4b) to (3.3.4e) can be written as

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0 \tag{3.3.5a}$$

$$c_i(\mathbf{x}^*) = 0 \quad i = 1, \ldots, m_{\mathcal{E}} \tag{3.3.5b}$$

$$c_i(\mathbf{x}^*) \leq 0 \quad i = m_{\mathcal{E}} + 1, \ldots, m \tag{3.3.5c}$$

$$\lambda_i \geq 0 \quad i = m_{\mathcal{E}} + 1, \ldots, m \tag{3.3.5d}$$

$$\lambda_i = 0 \quad i \notin \mathcal{A}(\mathbf{x}^*) \tag{3.3.5e}$$

Example 3.3.5 (IQEX1 continued) *For the problem defined in Example 3.3.3*

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -2x_1 \\ -2(x_2 - 1)(x_2 - 3)^2 - 2(x_2 - 1)^2(x_2 - 3) - \frac{1}{2} \end{bmatrix},$$

$$\nabla c_1(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ -1 \end{bmatrix}, \nabla c_2(\mathbf{x}) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \nabla c_3(\mathbf{x}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

At $\tilde{\mathbf{x}} = [\,0 \quad 2\,]^T$ *the active constraint set is* $\mathcal{A}(\tilde{\mathbf{x}}) = \{2\}$ *so equation (3.3.5a) is*

$$\nabla f(\tilde{\mathbf{x}}) + \bar{\lambda}_2 \nabla c_2(\tilde{\mathbf{x}}) = 0 \Longrightarrow \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} + \bar{\lambda}_2 \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which cannot be satisfied for any value of $\bar{\lambda}_2$ *(the first equation requires* $\bar{\lambda}_2 = 0$*, while the second equation requires that* $\bar{\lambda}_2 = \frac{1}{2}$*). Hence* $\tilde{\mathbf{x}}$ *does not satisfy the the KKT conditions.*

At $\overline{\mathbf{x}} = [\,0 \quad 0\,]^T$ *the active constraint set is* $\mathcal{A}(\overline{\mathbf{x}}) = \{1\}$ *so equation (3.3.5a) is*

$$\nabla f(\overline{\mathbf{x}}) + \bar{\lambda}_1 \nabla c_1(\overline{\mathbf{x}}) = 0 \Longrightarrow \begin{bmatrix} 0 \\ \frac{47}{2} \end{bmatrix} + \bar{\lambda}_1 \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

giving $\bar{\lambda}_1 = \frac{47}{2}$. *As the constraints* $c_2(\overline{\mathbf{x}}) < 0$ *and* $c_3(\overline{\mathbf{x}}) < 0$ *are not active the corresponding Lagrange multipliers are* $\bar{\lambda}_2 = 0$ *and* $\bar{\lambda}_3 = 0$. *Thus* $\overline{\mathbf{x}}$, *where* $f(\overline{\mathbf{x}}) = -9$, *satisfies the first order necessary conditions for a minimizer.*

At $\hat{\mathbf{x}} = [\,1 \quad 3\,]^T$ *the active constraint set is* $\mathcal{A}(\overline{\mathbf{x}}) = \{2,3\}$ *so equation (3.3.5a) is*

$$\nabla f(\hat{\mathbf{x}}) + \hat{\lambda}_2 \nabla c_2(\hat{\mathbf{x}}) + \hat{\lambda}_3 \nabla c_3(\hat{\mathbf{x}}) = 0 \Longrightarrow \begin{bmatrix} -2 \\ -\frac{1}{2} \end{bmatrix} + \hat{\lambda}_2 \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \hat{\lambda}_3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

giving $\hat{\lambda}_2 = -\frac{3}{4}$ *and* $\hat{\lambda}_3 = \frac{5}{4}$. *As there is a negative multiplier* $\hat{\lambda}_2$ *corresponding to an active inequality constraint the point* $\hat{\mathbf{x}}$ *is not a local minimizer. Similarly as there is a positive multiplier* $\hat{\lambda}_3$ *corresponding to an active inequality constraint the point* $\hat{\mathbf{x}}$ *is not a local maximizer. Thus* $\hat{\mathbf{x}}$, *where* $f(\hat{\mathbf{x}}) = -\frac{5}{2}$, *is neither a local minimizer nor a local maximizer.*

The condition that $\mathbf{x}^*$ is a regular point guarantees the existence and uniqueness of the multipliers λ^* satisfying (3.3.5a) at a local minimizer. There are other weaker conditions which guarantee the existence, but not uniqueness, of the multipliers satisfying (3.3.5a). Such conditions are called *constraint qualifications.*

Definition 3.3.6 *A feasible point* $\mathbf{x}$ *which is not regular is a* degenerate *point.*

For example if $t(\mathbf{x}) = |\mathcal{A}(\mathbf{x})| > n$ then $\mathbf{x}$ is degenerate as any set of more than n vectors in $\mathbb{R}^n$ is linearly dependent. An example in which the multipliers do not exist at a minimizer and an example in which there is an infinite set of multipliers are given in Exercise 4 in Section 3.3.4.

Definition 3.3.7 *A* constrained stationary point $\mathbf{x}^*$ *is a feasible point at which there exists multipliers* $\lambda_i^*, i \in \mathcal{A}(\mathbf{x}^*)$ *satisfying*

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0.$$

If $\lambda_i(\mathbf{x}^*) \geq 0$ for all $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ then $\mathbf{x}^*$ satisfies the first order necessary conditions for a minimizer. Thus if $\lambda_i^* < 0$ for an active inequality constraint $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ then $\mathbf{x}^*$ is *not* a local minimizer. If $\lambda_i(\mathbf{x}^*) \leq 0$ for all $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ then $\mathbf{x}^*$ satisfies the first order necessary conditions for a maximizer. This implies that if $\lambda_i^* > 0$ for an active inequality constraint $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ then $\mathbf{x}^*$ is *not* a local maximizer.

In general second order information is required to determine if a constrained stationary point is a local minimizer or local maximizer. However if $\mathbf{x}^*$ is a vertex of the feasible region then there are first order sufficient conditions.

Proposition 3.3.8 *Let* $\mathbf{x}^*$ *be a regular constrained stationary point with* $t(\mathbf{x}^*) = n$.

1. *If* $\lambda_i^* > 0$ *for all* $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ *then* $\mathbf{x}^*$ *is a strict local minimizer.*
2. *If* $\lambda_i^* < 0$ *for all* $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ *then* $\mathbf{x}^*$ *is a strict local maximizer.*
3. *If there exist* $i, j \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ *such that* $\lambda_i^* > 0$ *and* $\lambda_j^* < 0$ *then* $\mathbf{x}^*$ *is a constrained saddle point.*

In the unconstrained case ($\mathcal{E} = \emptyset, \mathcal{I} = \emptyset, m = 0$) and the equality constrained case ($\mathcal{I} = \emptyset, m = m_{\mathcal{E}}$) with $|\mathcal{A}(\mathbf{x})| < n$ so the feasible region is not an isolated point, there are no first order sufficient conditions.

Let $t(\mathbf{x}) = |\mathcal{A}(\mathbf{x})|$ be the number of active constraints at the point $\mathbf{x}$ and define the n by $t(\mathbf{x})$ matrix $A(\mathbf{x})$ by

$$A(\mathbf{x}) = \left[\ \nabla c_i(\mathbf{x}) \quad i \in \mathcal{A}(\mathbf{x})\ \right].$$

$A(\mathbf{x})$ is the matrix whose columns are the gradients of the active constraints at $\mathbf{x}$. A point $\mathbf{x}$ is regular if $A(\mathbf{x})$ has full rank. Let $\mu \in \mathbb{R}^{t(\mathbf{x})}$ be the vector of multipliers corresponding to the active constraints, so

$$\mu^T = \left[\ \lambda_i \quad i \in \mathcal{A}(\mathbf{x})\ \right].$$

Then (3.3.4b) and (3.3.5a) can be written at

$$\nabla f(\mathbf{x}^*) + A(\mathbf{x}^*)\mu^* = 0.$$

Example 3.3.9 (IQEX2) *Find all constrained stationary points for the objective function*

$$f(\mathbf{x}) = x_1^2 + x_2^2 - 6x_1 - 2x_2 + 10$$

and feasible region

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : -2 \le x_1 \le 2,\ 0 \le x_2 \le 4\}.$$

If possible identify them as local/global minimizers, local/global maximizers or constrained saddle points.

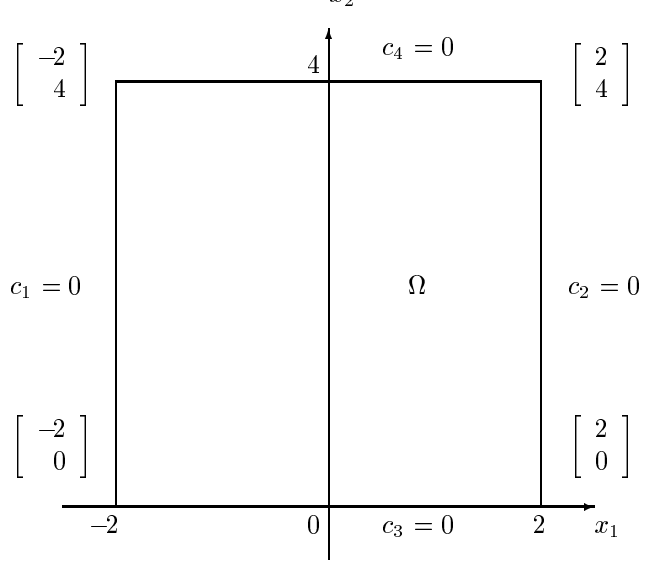


Figure 3.3.2: Feasible region Ω

A constrained stationary point satisfies

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0 \quad \textit{where} \quad \mathcal{A}(\mathbf{x}^*) = \{i \in 1, \ldots, m : c_i(\mathbf{x}^*) = 0\}.$$

For this problem $m_{\mathcal{E}} = 0$, $m = 4$, $\mathcal{I} = \{1, 2, 3, 4\}$,

$$c_1(\mathbf{x}) = -x_1 - 2 \quad c_2(\mathbf{x}) = x_1 - 2 \quad c_3(\mathbf{x}) = -x_2 \quad c_4(\mathbf{x}) = x_2 - 4$$

$$\nabla c_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \nabla c_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \nabla c_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \nabla c_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The objective function has

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 - 6 \\ 2x_2 - 2 \end{bmatrix} \qquad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

To find all constrained stationary points all the different possibilities for the set $\mathcal{A}(\mathbf{x}^*)$ *need to be considered. From the sketch of the feasible region in Figure 3.3.2 the only possibilities for* $\mathcal{A}(\mathbf{x}^*)$ *are*

$\mathcal{A}(\mathbf{x}^*)$	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,3\}$	$\{1,4\}$	$\{2,3\}$	$\{2,4\}$
t^*	0	1	1	1	1	2	2	2	2
A^*		$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Consider each possible active set in turn.

1. $\mathcal{A}(\mathbf{x}^*) = \emptyset$, $\nabla f(\mathbf{x}^*) = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$, $\mathbf{c}(\mathbf{x}^*) = [\,-5 \quad 1 \quad -1 \quad -3\,]^T \not\leq 0$, *so* $\mathbf{x}^*$ *is not feasible.*

2. $\mathcal{A}(\mathbf{x}^*) = \{1\}$, $\nabla f(\mathbf{x}^*) + \lambda_1 \nabla c_1(\mathbf{x}) = 0$, $c_1(\mathbf{x}) = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$, $\lambda_1^* = -10$, $f^* = 25$, $\mathbf{c}(\mathbf{x}^*) = [\,0 \quad -4 \quad -1 \quad -3\,]^T \leq 0$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As* $\lambda_1^* < 0$ *this point is not a local minimizer.*

3. $\mathcal{A}(\mathbf{x}^*) = \{2\}$, $\nabla f(\mathbf{x}^*) + \lambda_2 \nabla c_2(\mathbf{x}) = 0$, $c_2(\mathbf{x}) = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\lambda_2^* = 2$, $f^* = 1$, $\mathbf{c}(\mathbf{x}^*) = [\,-4 \quad 0 \quad -1 \quad -3\,]^T \leq 0$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As* $\lambda_2^* > 0$ *this point is not a local maximizer.*

4. $\mathcal{A}(\mathbf{x}^*) = \{3\}$, $\nabla f(\mathbf{x}^*) + \lambda_3 \nabla c_3(\mathbf{x}) = 0$, $c_3(\mathbf{x}) = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$, $\lambda_3^* = -2$, $\mathbf{c}(\mathbf{x}^*) = [\,-5 \quad 1 \quad 0 \quad -4\,]^T \not\leq 0$, *so* $\mathbf{x}^*$ *is not a feasible point.*

5. $\mathcal{A}(\mathbf{x}^*) = \{4\}$, $\nabla f(\mathbf{x}^*) + \lambda_4 \nabla c_4(\mathbf{x}) = 0$, $c_4(\mathbf{x}) = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\lambda_4^* = -6$, $\mathbf{c}(\mathbf{x}^*) = [\,-5 \quad 1 \quad -4 \quad 0\,]^T \not\leq 0$, *so* $\mathbf{x}^*$ *is not a feasible point.*

6. $\mathcal{A}(\mathbf{x}^*) = \{1,3\}$, $c_1^* = 0, c_3^* = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$, $\mathbf{c}(\mathbf{x}^*) = [\,0 \quad -4 \quad 0 \quad -4\,]^T \leq 0$, $f^* = 26$, $\nabla f(\mathbf{x}^*) + \lambda_1 \nabla c_1(\mathbf{x}) + \lambda_3 \nabla c_3(\mathbf{x}) = 0 \Longrightarrow \lambda_1^* = -10, \lambda_3^* = -2$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As* $\mathbf{x}^*$ *is a regular point with* $|\mathcal{A}(\mathbf{x}^*)| = n$ *and* $\lambda_i^* < 0$ *for all inequality constraints,* $\mathbf{x}^*$ *is a strict local maximizer.*

7. $\mathcal{A}(\mathbf{x}^*) = \{1,4\}$, $c_1^* = 0, c_4^* = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$, $\mathbf{c}(\mathbf{x}^*) = [\,0 \quad -4 \quad -4 \quad 0\,]^T \leq 0$, $f^* = 34$, $\nabla f(\mathbf{x}^*) + \lambda_1 \nabla c_1(\mathbf{x}) + \lambda_4 \nabla c_4(\mathbf{x}) = 0 \Longrightarrow \lambda_1^* = -10, \lambda_4^* = -6$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As* $\mathbf{x}^*$ *is a regular point with* $|\mathcal{A}(\mathbf{x}^*)| = n$ *and* $\lambda_i^* < 0$ *for all inequality constraints,* $\mathbf{x}^*$ *is a strict local maximizer.*

8. $\mathcal{A}(\mathbf{x}^*) = \{2,3\}$, $c_2^* = 0, c_3^* = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, $\mathbf{c}(\mathbf{x}^*) = [\,-4 \quad 0 \quad 0 \quad -4\,]^T \leq 0$, $f^* = 26$, $\nabla f(\mathbf{x}^*) + \lambda_2 \nabla c_2(\mathbf{x}) + \lambda_3 \nabla c_3(\mathbf{x}) = 0 \Longrightarrow \lambda_2^* = 2, \lambda_3^* = -2$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As there exist an inequality constraints with a positive multiplier and an inequality constraint with a negative multiplier* $\mathbf{x}^*$ *is neither a minimizer not a maximizer.*

9. $\mathcal{A}(\mathbf{x}^*) = \{2,4\}$, $c_2^* = 0, c_4^* = 0 \Longrightarrow \mathbf{x}^* = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$, $\mathbf{c}(\mathbf{x}^*) = [\,-4 \quad 0 \quad -4 \quad 0\,]^T \leq 0$, $f^* = 10$, $\nabla f(\mathbf{x}^*) + \lambda_2 \nabla c_2(\mathbf{x}) + \lambda_4 \nabla c_4(\mathbf{x}) = 0 \Longrightarrow \lambda_2^* = 2, \lambda_4^* = -6$, *so* $\mathbf{x}^*$ *is a feasible constrained stationary point. As there exist an inequality constraints with a positive multiplier and an inequality constraint with a negative multiplier* $\mathbf{x}^*$ *is neither a minimizer not a maximizer.*

The feasible region Ω *is the intersection of half-spaces, so is convex. The Hessian* $\nabla^2 f(\mathbf{x})$ *of the objective is positive definite for all* $\mathbf{x}$*, so* f *is strictly convex on* Ω*. Hence any constrained stationary point with* $\lambda_i^* \geq 0, i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$ *is the unique global minimizer, and local maximizers occur at extreme points of* Ω*. Thus* $\mathbf{x}^* = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ *is the unique global minimizer,* $\lambda_2^* = 2$*, and the global minimum is* $f^* = 1$ *The feasible region is closed and bounded, and the objective function is continuous, so global extrema are guaranteed to exist for this problem. Hence the extreme point(s) with the largest function value are the global maximizer(s). In this case the global maximum is* $f^* = 34$ *with global maximizer* $\mathbf{x}^* = \begin{bmatrix} -2 \\ 4 \end{bmatrix}$.

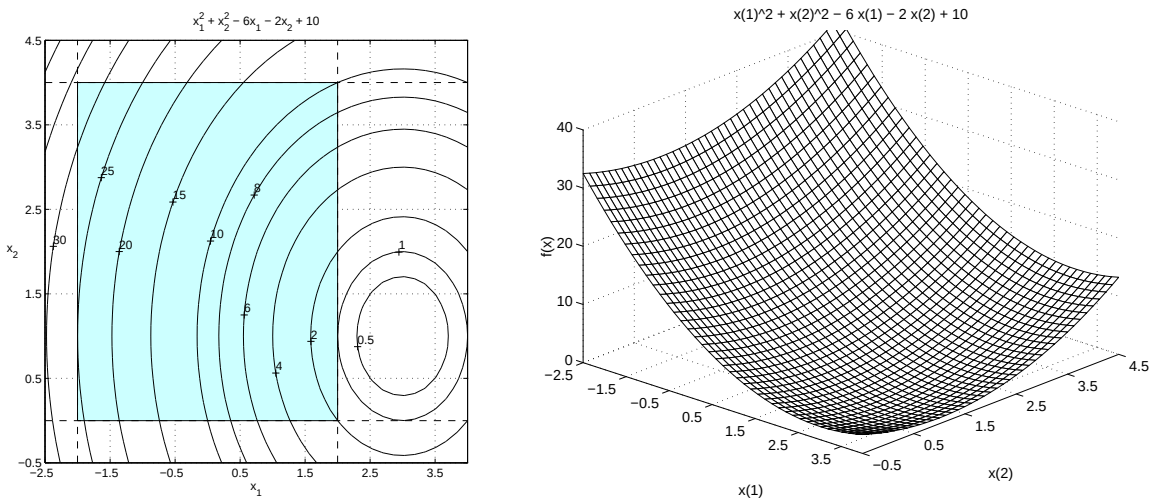


Figure 3.3.3: Contour and surface plot of f on Ω

3.3.2 Second order conditions

Proposition 3.3.10 (Second order sufficient conditions) *Let $f, c_i \in C^2(\Omega)$ and let $\mathcal{A}(\mathbf{x}^*) = \{i \in \mathcal{E} \cup \mathcal{I} : c_i(\mathbf{x}^*) = 0\}$ bet the set of active constraints at $\mathbf{x}^*$. If there exists multipliers $\lambda^* \in \mathbb{R}^m$ such that*

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0 \tag{3.3.6a}$$

$$c_i(\mathbf{x}^*) = 0 \quad i \in \mathcal{E}, \tag{3.3.6b}$$

$$c_i(\mathbf{x}^*) \leq 0 \quad i \in \mathcal{I}, \tag{3.3.6c}$$

$$\lambda_i^* = 0 \quad i \notin \mathcal{A}(\mathbf{x}^*), \tag{3.3.6d}$$

$$\lambda_i^* \geq 0 \quad i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}, \tag{3.3.6e}$$

and

$$\mathbf{d}^T \nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}^*, \lambda^*) \mathbf{d} > 0 \quad \textit{for all } \mathbf{d} \in \mathcal{F}^*$$

where

$$\mathcal{F}^* = \left\{ \mathbf{d} \in \mathbb{R}^n : \begin{array}{ll} \mathbf{d}^T \nabla c_i(\mathbf{x}^*) = 0 & i \in \mathcal{E} \cup \{i \in \mathcal{I} : \lambda_i^* > 0\} \\ \mathbf{d}^T \nabla c_i(\mathbf{x}^*) \leq 0 & i \in \{i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I} : \lambda_i^* = 0\} \end{array} \right\}.$$

then $\mathbf{x}^*$ is a strict local minimizer of f on Ω.

See Fletcher [Fle87] for a proof.

For a problem with inequality constraints the complementarity condition and feasibility conditions are

$$\lambda_i^* c_i(\mathbf{x}^*) = 0 \quad i \in \mathcal{E} \cup \mathcal{I}, \qquad \lambda_i^* \geq 0 \quad i \in \mathcal{I}, \qquad c_i(\mathbf{x}^*) \leq 0 \quad i \in \mathcal{I}.$$

At a feasible point which is *not active* $c_i(\mathbf{x}^*) < 0$, which implies that $\lambda_i^* = 0$. Hence for any feasible KKT point $\lambda_i^* = 0$ for $i \notin \mathcal{A}(\mathbf{x}^*)$, so the Lagrange multipliers satisfy

$$\lambda_i^* = 0 \quad i \notin \mathcal{A}(\mathbf{x}^*), \tag{3.3.7a}$$

$$\lambda_i^* \geq 0 \quad i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}. \tag{3.3.7b}$$

Definition 3.3.11 *The constrained stationary point $\mathbf{x}^*, \lambda^*$ satisfies* strict complementarity *if*

$$\lambda_i^* > 0 \quad \forall\, i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}. \tag{3.3.8}$$

Strict complementarity implies

$$\mathcal{F}^* = \left\{ \mathbf{d} \in \mathbb{R}^n : \mathbf{d}^T \nabla c_i(\mathbf{x}^*) = 0 \; \forall \, i \in \mathcal{A}(\mathbf{x}^*) \right\}.$$

Let $\mathbf{x}^*$ be a regular point and let $t^* = |\mathcal{A}(\mathbf{x}^*)|$. Let $Z^* \in \mathbb{R}^{n \times n - t*}$ be a full rank matrix satisfying ${Z^*}^T \nabla c_i(\mathbf{x}^*) = 0$, so the columns of Z^* form a basis for the tangent space to the active constraints at $\mathbf{x}^*$. The *reduced Hessian* of the Lagrangian is

$$W_Z^* = {Z^*}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) Z^*. \tag{3.3.9}$$

Let $\mathbf{x}^*, \lambda^*$ be a regular constrained stationary point.

1. If $\lambda_i^* > 0$ for all $i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*)$ and W_Z^* positive definite then $\mathbf{x}^*$ is a strict local minimizer.
2. If $\lambda_i^* < 0$ for all $i \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*)$ and W_Z^* negative definite then $\mathbf{x}^*$ is a strict local maximizer.
3. (a) If there are $i, j \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*)$ such that $\lambda_i^* > 0$ and $\lambda_j^* < 0$, or
   (b) If there is a $j \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*)$ such that $\lambda_j^* > 0$ and W_Z^* has a negative eigenvalue, or
   (c) if there is a $j \in \mathcal{I} \cap \mathcal{A}(\mathbf{x}^*)$ such that $\lambda_j^* < 0$ and W_Z^* has a positive eigenvalue, or
   (d) if W_Z^* is indefinite,

   then $\mathbf{x}^*$ is neither a local minimizer nor a local maximizer.

Example 3.3.12 (IQEX1 continued) *Consider the point $\overline{\mathbf{x}} = 0$ in Example 3.3.3. The active constraints at $\overline{\mathbf{x}}$ are $\mathcal{A}(\overline{\mathbf{x}}) = \{1\}$ and $\bar{\lambda}_1 = \frac{47}{2}$. The Hessian of the objective function is*

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} -2 & 0 \\ 0 & -2(x_2-3)^2 - 8(x_2-1)(x_2-3) - 2(x_2-1)^2 \end{bmatrix}$$

and the Hessian of the active constraint is

$$\nabla^2 c_1(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Hence the Hessian of the Lagrangian at $\overline{\mathbf{x}}$ is

$$\bar{W} \equiv \nabla^2 \mathcal{L}(\overline{\mathbf{x}}, \bar{\lambda}) = \nabla^2 f(\overline{\mathbf{x}}) - \bar{\lambda}_1 \nabla^2 c_1(\overline{\mathbf{x}}) = \begin{bmatrix} -2 & 0 \\ 0 & -44 \end{bmatrix} + \frac{47}{2} \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 45 & 0 \\ 0 & -44 \end{bmatrix}.$$

Clearly $\bar{W}$ is indefinite. As $\bar{\lambda}_1 = \frac{47}{2} > 0$ the point $\overline{\mathbf{x}}$ satisfies strict complementarity, so $\bar{\mathcal{F}}$ is the tangent space to the active constraints at $\overline{\mathbf{x}}$. A suitable basis matrix is

$$\bar{Z} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Longrightarrow W_Z(\overline{\mathbf{x}}, \bar{\lambda}) = \bar{Z}^T \nabla^2 \mathcal{L}(\overline{\mathbf{x}}, \bar{\lambda}) \bar{Z} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 45 & 0 \\ 0 & -44 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 45.$$

As $\bar{\lambda}_1 > 0$ and the reduced Hessian of the Lagrangian $W_Z(\overline{\mathbf{x}}, \bar{\lambda})$ is positive definite, the point $\overline{\mathbf{x}}$ is a strict local minimizer.

Example 3.3.13 (IQEX2 continued) *In Example 3.3.9 consider the two cases where first order information is not sufficient to determine the nature of the constrained stationary point.*

1. $\mathcal{A}(\mathbf{x}^*) = \{1\}$, $\mathbf{x}^* = \begin{bmatrix} -2 \\ 1 \end{bmatrix}$, $\lambda_1^* = -10$, $f^* = 25$.

$$\nabla c_1^* = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \Longrightarrow Z^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad W^* = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \Longrightarrow W_Z^* = {Z^*}^T W^* Z^* = 2.$$

As $\lambda_1^* < 0$ and $W_Z^* > 0$ this point is neither a local minimizer nor a local maximizer.

2. $\mathcal{A}(\mathbf{x}^*) = \{2\}$, $\mathbf{x}^* = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $\lambda_2^* = 2$, $f^* = 1$.

$$\nabla c_1^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Longrightarrow Z^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad W^* = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \Longrightarrow W_Z^* = Z^{*T} W^* Z^* = 2.$$

As $\lambda_2^* > 0$ *and* $W_Z^* > 0$ *this point is a strict local minimizer.*

In both cases the fact that W^* *is positive definite implies* W_Z^* *is positive definite for any* Z^**, so the matrices* Z^*, W_Z^* *need not have been explicitly calculated.*

Example 3.3.14 (IQEX3) *Find all constrained stationary points of the problem*

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^2}{Minimize} & f(\mathbf{x}) = x_1^2 + x_2^2 + 3x_2 \\ Subject\ to & x_1^2 + (x_2+1)^2 \geq 1 \end{array}$$

and, if possible, identify them as local minimizers, local maximizers, etc.

In standard form the feasible region is

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) = -x_1^2 - (x_2+1)^2 + 1 \leq 0\}.$$

In Figure 3.3.4 the feasible region Ω *is the boundary and exterior of the circle with centre* $[\ 0 \quad -1\ ]^T$ *and radius 1. Hence* Ω *is not a convex set, so convexity cannot be used to identify local/global minimizers.*

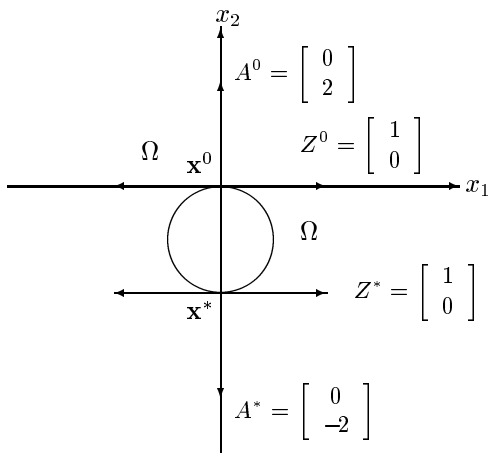


Figure 3.3.4: Feasible region Ω

(The objective function is convex on $\mathbb{R}^2$*, but this is irrelevant as the feasible region* Ω *is not convex). Also as* Ω *is unbounded there is no guarantee global extrema exist.*

For this problem $m_{\mathcal{E}} = 0$, $m = 1$ *and*

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 + 3 \end{bmatrix}, \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \nabla c_1(\mathbf{x}) = \begin{bmatrix} -2x_1 \\ -2(x_2+1) \end{bmatrix}, \nabla^2 c_1(\mathbf{x}) = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}.$$

The possibilities for the set of active constraints $\mathcal{A}(\mathbf{x}^*)$ *are* $\emptyset$ *and* $\{1\}$*. If* $\mathcal{A}(\overline{\mathbf{x}}) = \emptyset$ *then* $\nabla f(\overline{\mathbf{x}}) = 0$ *implies* $\overline{\mathbf{x}} = [\ 0 \quad -\frac{3}{2}\ ]^T$*. As* $c_1(\overline{\mathbf{x}}) = \frac{3}{4} > 0$ *the point* $\overline{\mathbf{x}}$ *is not feasible. If* $\mathcal{A}(\mathbf{x}^*) = \{1\}$ *then*

$$\nabla f(\mathbf{x}^*) + \lambda_1 \nabla c_1(\mathbf{x}^*) = 0 \Longrightarrow \begin{bmatrix} 2x_1 \\ 2x_2 + 3 \end{bmatrix} + \lambda_1 \begin{bmatrix} -2x_1 \\ -2x_2 - 2 \end{bmatrix} = 0 \tag{3.3.10}$$

and

$$c_1(\mathbf{x}^*) = 0 \Longrightarrow -x_1^2 - (x_2+1)^2 + 1 = 0. \tag{3.3.11}$$

The first component of (3.3.10) implies $x_1^* = 0$ *(as* $\lambda_1 = 1$ *means that the second component cannot be zero). Substituting* $x_1 = 0$ *in (3.3.11) gives* $(x_2+1)^2 = 1$ *so either* $x_2 = 0$ *or* $x_2 = -2$. *Thus the constrained stationary points are*

$$\mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \lambda_1^0 = \frac{3}{2}, f(\mathbf{x}^0) = 0 \quad \textit{and} \quad \mathbf{x}^* = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \lambda_1^* = \frac{1}{2}, f(\mathbf{x}^*) = -2.$$

As the number of active constraints is less than the number of variables these conditions are no sufficient to determine the nature of these constrained stationary points. In each case the multiplier for the inequality constraint is positive, so neither point is a local maximizer.

The only non-regular point is $\hat{\mathbf{x}} = [\,0 \quad -1\,]^T$ *where* $\nabla c_1(\hat{\mathbf{x}}) = 0$. *However* $c_1(\hat{\mathbf{x}}) = 1 > 0$ *so* $\hat{\mathbf{x}}$ *is not feasible.*

The second order conditions require bases for the tangent spaces to the constraint at $\mathbf{x}^0$ *and* $\mathbf{x}^*$. *Let* $t(\mathbf{x}) = |\mathcal{A}(\mathbf{x})|$. *Then* $Z(\mathbf{x}) \in \mathbb{R}^{n \times n - t(\mathbf{x})}$ *must have linearly independent columns and satisfy* $A(\mathbf{x})^T Z(\mathbf{x}) = 0$.

$$A(\mathbf{x}^0) = \begin{bmatrix} 0 \\ -2 \end{bmatrix} \Longrightarrow Z^0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$A(\mathbf{x}^*) = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \Longrightarrow Z^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

More generally any point $\mathbf{x}$ *satisfying* $c_1(\mathbf{x}) = 0$ *has*

$$A(\mathbf{x}) = \begin{bmatrix} -2x_1 \\ -2(x_2+1) \end{bmatrix} \Longrightarrow Z(x) = \begin{bmatrix} x_2+1 \\ -x_1 \end{bmatrix}.$$

The Hessian of the Lagrangian function is

$$W(\mathbf{x}, \lambda) = \nabla^2 f(\mathbf{x}) + \sum_{i \in \mathcal{A}(\mathbf{x})} \lambda_i \nabla^2 c_i(\mathbf{x}) = \begin{bmatrix} 2 - 2\lambda_1 & 0 \\ 0 & 2 - 2\lambda_1 \end{bmatrix} = 2(1-\lambda_1)I.$$

Hence the reduced Hessian is

$$W_R(\mathbf{x}, \lambda) = Z(\mathbf{x})^T W(\mathbf{x}, \lambda) Z(\mathbf{x}) = 2(1-\lambda_1)(x_1^2 + (x_2+1)^2).$$

At $\mathbf{x}^0 = [\,0 \quad 0\,]^T, \lambda_1^0 = \frac{3}{2}$ *the reduced Hessian of the Lagrangian is* $W_R(\mathbf{x}^0, \lambda^0) = -1 < 0$, *so* $\mathbf{x}^0$ *is neither a minimizer not a maximizer. At* $\mathbf{x}^* = [\,0 \quad -2\,]^T, \lambda_1^* = \frac{1}{2}$ *and* $W_R(\mathbf{x}^*, \lambda^*) = 1 > 0$, *so* $\mathbf{x}^*$ *is a strict local minimizer.*

Even though the Hessian $\nabla^2 f(\mathbf{x})$ *is positive definite for all* $\mathbf{x}$ *the point* $\mathbf{x}^0$ *is not a local minimizer at the function decreases when moving around the constraint boundary. This illustrates the fact that the constraint curvature must be taken into account through the Hessian of the Lagrangian function.*

3.3.3 Sensitivity analysis

Sensitivity results typically require that the solution $\mathbf{x}^*$ is "nice" in some sense, for example

1. Linear independence: the gradients $\nabla c_i(\mathbf{x}^*)$ for the active constraints $i \in \mathcal{A}(\mathbf{x}^*)$ are linear independent.

2. Strict complementarity: the Lagrange multiplier $\lambda_i^* > 0$ for $i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}$

3. Second order sufficiency: $\mathbf{d}^T \nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}^*, \lambda^*)\mathbf{d} > 0$ for all non-zero $\mathbf{d} \in \mathcal{F}^*$.

If all these conditions are satisfied then one can get expressions for the derivatives of the objective value and solution point not only with respect to perturbations of the right-hand-sides of the constraints, but also with respect to more general perturbations of the objective and constraint functions. On the other hand if not all these conditions are satisfied then the perturbed function value may not be a differentiable function of the perturbations. A good introduction to sensitivity results is Fiacco [Fia83].

The Lagrange multiplier λ_i^* gives a first order estimate of the rate of change of the objective function with respect to changes in the ith constraint. Thus if a constraint $c_i(\mathbf{x}) \leq 0$ becomes $c_i(\mathbf{x}) + \epsilon_i \leq 0$ then the change in the optimal objective is

$$\Delta f^* \approx \lambda_i^* \epsilon_i$$

and an estimate of the new optimal objective value is

$$f^*(\epsilon_i) \approx f(\mathbf{x}^*) + \Delta f^* = f(\mathbf{x}^*) + \lambda_i^* \epsilon_i.$$

If the ith inequality constraint is not active at $\mathbf{x}^*$, so $i \notin \mathcal{A}(\mathbf{x}^*)$, then $\lambda_i^* = 0$. In this case the constant term in the i constraint can be changed a little without affecting the optimal solution.

Example 3.3.15 (IQEX4) *Consider the problem*

$$\begin{array}{ll} \textit{Minimize} & f(\mathbf{x}) = (2x_1 - 3)^2 + (2x_2 - 3)^2 \\ \mathbf{x} \in \mathbb{R}^2 & \\ \textit{Subject to} & c_1(\mathbf{x}) \equiv x_1^2 + x_2^2 - 2 \leq 0 \\ & c_2(\mathbf{x}) \equiv -x_1 \leq 0 \\ & c_3(\mathbf{x}) \equiv -x_2 \leq 0. \end{array}$$

The point $\mathbf{x}^* = [\,1 \quad 1\,]^T$, *has* $c(\mathbf{x}^*) = [\,0 \quad -1 \quad -1\,]^T \leq 0$, *so* $\mathbf{x}^*$ *is feasible and the set of active*

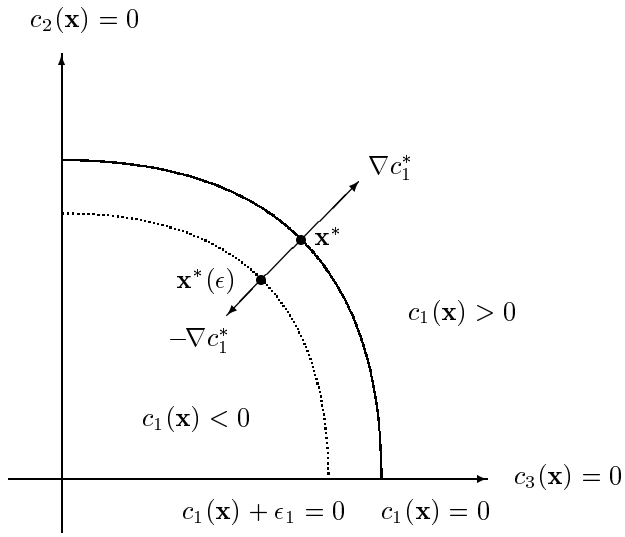


Figure 3.3.5: Perturbation of constraints

constraints is $\mathcal{A}(\mathbf{x}^*) = \{1\}$. *The point* $\mathbf{x}^*$ *is a constrained stationary point with* $\lambda_1^* = 2$ *as*

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla c_i(\mathbf{x}^*) = \begin{bmatrix} -4 \\ -4 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The objective value at $\mathbf{x}^*$ *is* $f(\mathbf{x}^*) = 2$. *Let the first constraint be changed to* $x_1^2 + x_2^2 \leq 1.6$, *corresponding to the perturbation* $\epsilon_1 = 0.4$. *This is drawn in Figure 3.3.5. An estimate of the change in the optimal objective value is*

$$\Delta f^* \approx \lambda_1^* \epsilon_1 = 2 \times (0.4) = 0.8.$$

With the perturbed constraint $c_1(\mathbf{x}) + \epsilon \leq 0$ *active the optimal point is a solution of*

$$\begin{array}{ll} \nabla f(\mathbf{x}) + \lambda_1 \nabla c_1(\mathbf{x}) = 0 & \\ & \Longrightarrow \begin{bmatrix} 4(2x_1 - 3) \\ 4(2x_2 - 3) \end{bmatrix} + \lambda_1 \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ c_1(\mathbf{x}) + \epsilon = 0 & \\ & x_1^2 + x_2^2 - 2 + \epsilon = 0 \end{array}$$

giving

$$\mathbf{x}^*(\epsilon) = \sqrt{1 - \frac{\epsilon}{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \lambda_1^*(\epsilon) = -4 + \frac{6}{\sqrt{1 - \frac{\epsilon}{2}}}$$

with

$$f(\mathbf{x}^*(\epsilon)) = 2\left(2\sqrt{1 - \frac{\epsilon}{2}} - 3\right)^2 = 26 - 4\epsilon - 12\sqrt{4 - 2\epsilon}.$$

It is readily verified that $\mathbf{x}^*(0) = \mathbf{x}^*$ *and* $f(\mathbf{x}^*(0)) = f(\mathbf{x}^*) = 2$. *The actual change in the optimal objective value with* $\epsilon = 0.4$ *is*

$$f(\mathbf{x}^*(0.4)) - f(\mathbf{x}^*) = 26 - 4(0.4) - 12\sqrt{4 - 2(0.4)} - 2 = 0.9337472,$$

which compares with the first order estimate of 0.8. *The first derivative of the change in objective value is*

$$\frac{df(\mathbf{x}^*(\epsilon))}{d\epsilon} = -4 + \frac{12}{\sqrt{4 - 2\epsilon}} = \lambda_1^*(\epsilon) \Longrightarrow \left.\frac{df(\mathbf{x}^*(\epsilon))}{d\epsilon}\right|_{\epsilon=0} = 2 = \lambda_1^*.$$

The optimal value $f(\mathbf{x}^*(\epsilon))$ *of the perturbed problem and the linear approximation* $f(\mathbf{x}^*) + \lambda_1^* \epsilon_1$ *are sketched in Figure 3.3.6 As the constraints* $c_2(\mathbf{x}) = x_1 \leq 0$ *and* $c_3(\mathbf{x}) = -x_2 \leq 0$ *are not active at* $\mathbf{x}^*$ *the*

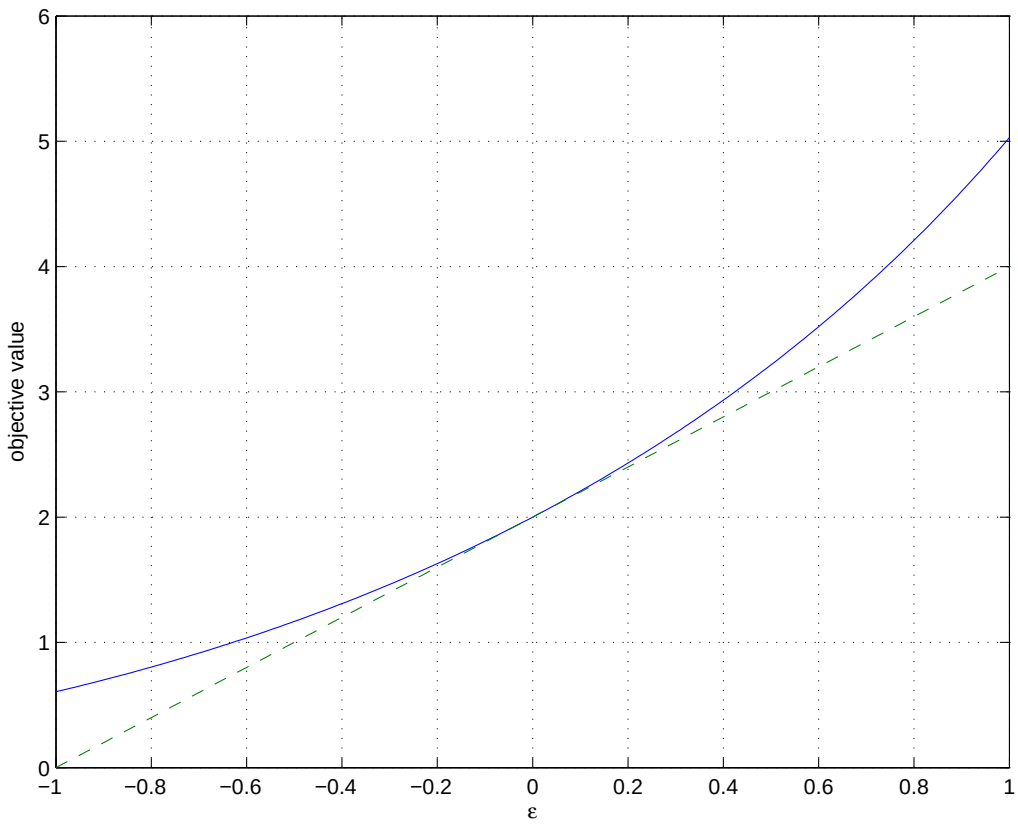


Figure 3.3.6: Optimal value $f(\mathbf{x}^*(\epsilon))$ and linear approximation $f(\mathbf{x}^*) + \lambda_1^* \epsilon$

corresponding Lagrange multipliers are $\lambda_2^* = 0$ *and* $\lambda_3^* = 0$. *Perturbing the constraint* $c_2(\mathbf{x}) = -x_1 + \epsilon_2 \leq 0$ *for any* $\epsilon_2 < 1$ *does not affect the optimal solution. Similarly the perturbed constraint* $c_3(\mathbf{x}) = -x_2 + \epsilon_3 \leq 0$ *does not change the optimal solution for any* $\epsilon_3 < 1$.

3.3.4 Exercises

1. Consider the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^2}{Minimize} & f(\mathbf{x}) = x_1^2 + 10x_2^2 - 5x_1 - x_2 + 13 \\ Subject\ to & c_1(\mathbf{x}) \equiv -x_1 \leq 0 \\ & c_2(\mathbf{x}) \equiv x_1 + x_2 - 2 \leq 0. \end{array}$$

   (a) Sketch the feasible region $\Omega = \{ \mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) \leq 0,\ c_2(\mathbf{x}) \leq 0 \}$.

   (b) Do global extrema of $f(\mathbf{x})$ on Ω exist?

   (c) Is this a convex or concave programming problem, and if so what does that tell you about solution(s) to the problem?

   (d) Find all the constrained stationary points, and identify them as far as possible using only first order information.

   (e) Where required use second order information to determine, if possible, the nature of the constrained stationary point(s).

2. Repeat all parts of Question 1 for the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^2}{Minimize} & f(\mathbf{x}) = x_1^2 + 10x_2^2 - 5x_1 - x_2 + 13 \\ Subject\ to & c_1(\mathbf{x}) \equiv -x_1^2 + x_2 = 0 \\ & c_2(\mathbf{x}) \equiv -x_1 \leq 0 \\ & c_3(\mathbf{x}) \equiv x_1 + x_2 - 2 \leq 0. \end{array}$$

   In this problem the constraints c_1 and c_2 from Question 1 have been relabelled constraints c_2 and c_3 respectively, and the equality constraint $c_1(\mathbf{x}) = 0$ has been added. The objective function $f(\mathbf{x})$ is unchanged.

3. A silo of fixed volume $V > 0$ is to be constructed. The silo consists of a cylinder topped by a hemisphere. Safety regulations state that the height of the cylinder must not exceed three times the radius of the cylinder. The cost of construction of the cylinder is \$100 per square metre, while the cost per square metre of construction of the hemisphere is twice that of the cylinder. The earth will provide a free floor. What are the dimensions of the silo that will minimize the cost of construction?

   (a) Pose this problem as a mathematical optimization problem in standard form.

   (b) Sketch the feasible region when $V = 2000\pi/3\, m^3$. Are any of constraints redundant? [Note: this value for V is only for use in the sketch].

   (c) Do global extrema exist for this problem?

   (d) Is this a convex or concave programming problem?

   (e) Find all the constrained stationary points and identify them, as far as possible, using only first order information.

   (f) Verify that the second order sufficient conditions hold at the minimizer.

   (g) Suppose you wish to build a silo with a volume of 10^6 litres.

      i. Calculate the optimal dimensions, cost and Lagrange multipliers.

      ii. Using the Lagrange multipliers estimate the change in cost if the volume is increased by 10%, and compare this with the actual change.

4. [**H**] Consider the feasible region

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : c_1(\mathbf{x}) \equiv -x_1^3 + x_2 \leq 0, c_2(\mathbf{x}) \equiv -x_2 \leq 0\}.$$

and the point $\mathbf{x}^* = [\,0 \quad 0\,]^T$.

(a) Show that $\mathbf{x}^*$ is feasible, but is not a regular point.

(b) Show that $\mathbf{x}^*$ is the minimizer of $f(\mathbf{x}) = x_1$ on Ω, but that there do not exist multipliers satisfying the KKT conditions.

(c) Show that $\mathbf{x}^*$ is the minimizer of $f(\mathbf{x}) = x_2$ on Ω, and find the set of *all* multipliers satisfying the KKT conditions.

5. [**H**] Consider the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^3}{Minimize} & f(\mathbf{x}) = -x_1^2 + x_2^2 - x_1 - x_2 - x_3 \\ Subject\ to & c_1(\mathbf{x}) = x_1 + x_2 + x_3 - 1 = 0 \\ & c_2(\mathbf{x}) = -x_1 \leq 0. \end{array}$$

and the point $\mathbf{x}^* = [\,0 \quad 0 \quad 1\,]^T$.

(a) Show that $\mathbf{x}^*$ is a regular constrained stationary point.

(b) Find a basis $Z(\mathbf{x}^*)$ for the tangent space to the active constraints at $\mathbf{x}^*$.

(c) Show that the reduced Hessian of the Lagrangian,

$$W_Z(\mathbf{x}^*, \lambda^*) = Z(\mathbf{x}^*)^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) Z(\mathbf{x}^*),$$

is positive definite.

(d) Find the set $\mathcal{F}^*$ of feasible directions with zero slope at $\mathbf{x}^*$,

$$\mathcal{F}^* = \left\{ \mathbf{d} \in \mathbb{R}^n : \begin{array}{ll} \mathbf{d}^T \nabla c_i(\mathbf{x}^*) = 0 & i \in \mathcal{E} \cup \{i \in \mathcal{I} \cap \mathcal{A}^* : \lambda_i^* > 0\} \\ \mathbf{d}^T \nabla c_i(\mathbf{x}^*) \leq 0 & i \in \{i \in \mathcal{I} \cap \mathcal{A}^* : \lambda_i^* = 0\} \end{array} \right\}.$$

(e) Show that there are directions $\mathbf{d} \in \mathcal{F}^*$ such that the necessary condition

$$\mathbf{d}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) \mathbf{d} \geq 0 \quad \text{for all } \mathbf{d} \in \mathcal{F}^*$$

for a local minimizer is not satisfied.

6. [**H**] Consider the problem of finding the global extrema of

$$f(\mathbf{x}) = x_1(1 + x_2)$$

on

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1 \geq x_2, x_1 \leq -x_2, x_2 \geq -1\}$$

(a) Sketch the feasible region Ω.

(b) Show that global extrema exist for this problem.

(c) Show that the feasible region Ω is convex, but that $f(\mathbf{x})$ is neither convex nor convcave.

(d) Show that this is an indefinite quadratic programming problem.

(e) Find all the possible sets of active constraintes, and all the constrained stationary points of $f(\mathbf{x})$ on Ω.

(f) Find the global minimizer and global maximizer of $f(\mathbf{x})$ on Ω.

(g) Consider the point $\overline{\mathbf{x}} = [\,0 \quad -1\,]^T$.

i. Show that $\overline{\mathbf{x}}$ is a constrained stationary point with the constraint $x_2 \geq -1$ active, but the Lagrange multiplier for this constraint is zero.

ii. Calculate the set of directions $\mathcal{F}(\overline{\mathbf{x}})$ and show that $\overline{\mathbf{x}}$ is neither a minimizer nor a maximizer of $f(\mathbf{x})$ on Ω.

Chapter 4

Methods for One-dimensional Problems

One-dimensional problems are important both in their own right, and because the are a key part of methods for multi-variable optimization. Hence it is essential that they are solved as efficiently and reliably as possible.

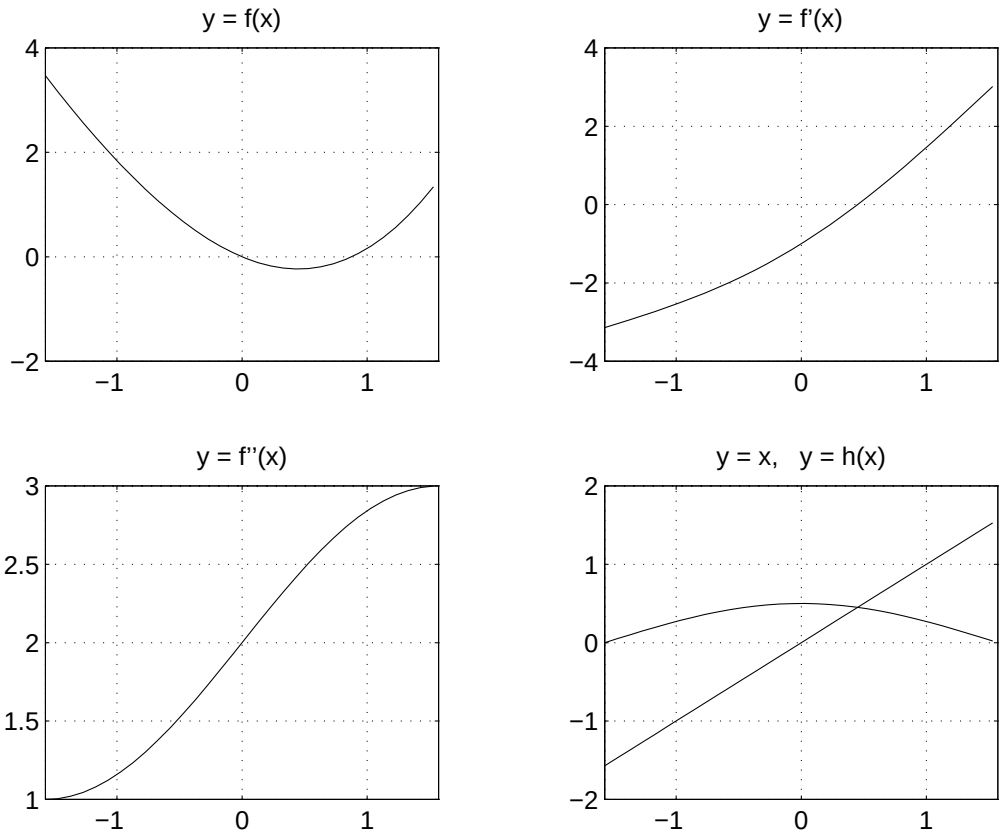


Figure 4.0.1: Function in Example 4.0.16

Example 4.0.16 *Consider*

$$f(x) = x^2 - sin(x) \qquad on \qquad \Omega = [\,-\frac{\pi}{2},\ \frac{\pi}{2}\,].$$

k	$x^{(k)}$	$f(x^{(k)})$	$f'(x^{(k)})$
1	.5	-.22942554	.12241744
2	.43879128	-.23230778	-.02768328
3	.45263292	-.23245827	.00596709
4	.44964938	-.23246523	-.00130080
5	.45029978	-.23246556	.00028289
6	.45015833	-.23246557	-.00006155
7	.45018911	-.23246558	.00001339
8	.45018242	-.23246558	-.00000291
9	.45018387	-.23246558	.00000063
10	.45018356	-.23246558	-.00000014
11	.45018362	-.23246558	.00000003
12	.45018361	-.23246558	-.00000000

Table 4.1.1: Fixed point iteration for $x = cos(x)/2$

Then

$$f'(x) = 2x - cos(x), \quad f''(x) = 2 + sin(x).$$

As the interval Ω is convex, and $f''(x) > 0$ for all x, f is a strictly convex function. Thus any stationary point of f in Ω is the global minimizer of f on Ω. Finding a stationary point requires the solution of $f'(x) = 0$, which requires a numerical method. A sketch of $cos(x)$ and $2x$ shows that there is exactly one stationary point in Ω at approximately $x = 0.5$. Rearranging $f'(x) = 0$ into the form $x = h(x)$ produces $x = cos(x)/2$.

4.1 Fixed point iteration

Fixed point iteration (or simple iteration) is the iterative method produced by taking $x^{(k+1)} = h(x^{(k)})$. If $x^{(1)}$ is sufficiently close to x^* and $|h'(x^*)| < 1$ then simple iteration produces a sequence $\{x^{(k)}\}$ which converges to x^*, and the rate of convergence is linear with rate $|h'(x^*)|$.

The results of fixed point iteration on Example 4.0.16 are displayed in Table 4.1 where all the numbers have been rounded to 8 decimal places and $x^{(12)}$ is x^* correct to this accuracy. As $h'(x^*) = -sin(x^*)/2 \approx -0.22$, the convergence is linear with rate about 0.22. The effects of different convergence criteria should also be noted. In the table the iteration was stopped when $|f'(x^{(k)})| < \frac{1}{2} \times 10^{-8}$. However if the convergence criterion was $|f(x^{(k+1)}) - f(x^{(k)})| < \frac{1}{2} \times 10^{-8}$ then the iteration would have stopped with $x^{(6)}$.

This example serves both to illustrate a simple numerical method, and the close connection between minimizing $f(x)$ and solving the $f'(x) = 0$. Note that it was because f was convex that the solution to $f'(x) = 0$ is known to be a minimizer. In general solutions to $f'(x) = 0$ can be maximizers, minimizers, or neither.

The basic idea of many numerical optimization techniques is to model the nonlinear function $f(x)$ at the point $x^{(k)}$ by a simpler function whose minimizer can easily be calculated. The minimizer of this model function is used to obtain a new estimate $x^{(k+1)}$ of a local minimizer x^* of $f(x)$. The choice of method depends upon the existence and computability of first and second derivatives of f.

4.2 Newton's Method

Suppose that f is twice continuously differentiable. Then at the point $x^{(k)}$ the function f can be modelled by the quadratic function $q_k(d)$ obtained from a second order Taylor series expansion about $x^{(k)}$, namely

$$f(x^{(k)} + d) \approx q_k(d) \equiv f(x^{(k)}) + f'(x^{(k)})\, d + \tfrac{1}{2} f''(x^{(k)})\, d^2.$$

k	$x^{(k)}$	$f(x^{(k)})$	$f'(x^{(k)})$	$f''(x^{(k)})$
1	.50000000	-.22942554	.12241744	2.4794255
2	.45062669	-.23246534	.00107905	2.4355298
3	.45018365	-.23246558	-.00000009	2.4351309
4	.45018361	-.23246558	-.00000000	2.4351309

Table 4.2.1: Newton's method on Example 1

The unconstrained minimizer $d^{(k)}$ of this quadratic model satisfies

$$q_k'(d) = f'(x^{(k)}) + f''(x^{(k)})\, d = 0,$$

giving

$$d^{(k)} = -\frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Note that $d^{(k)}$ is a minimizer of $q^{(k)}(d^{(k)})$ if and only if $f''(x^{(k)}) > 0$. Newton's method is obtained by using $d^{(k)}$ to provide a new estimate of a local minimizer, so

$$x^{(k+1)} = x^{(k)} + d^{(k)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})} \quad \text{when } f''(x^{(k)}) > 0.$$

Newton's method is not globally convergent as it is not defined at any point $x^{(k)}$ with $f''(x^{(k)}) \le 0$. However if the second order sufficient conditions for an unconstrained minimizer are satisfied at x^*, and $x^{(1)}$ is sufficiently close to x^* then Newton's method works very well, as Table 4.2.1 illustrates.

Proposition 4.2.1 *Let $f \in C^3(\mathbb{R})$ and let x^* be a local minimizer satisfying $f'(x^*) = 0$ and $f''(x^*) > 0$. If $x^{(1)}$ is sufficiently close to x^* then Newton's method produces a sequence of iterates $\{x^{(k)}\}$ such that $f''(x^{(k)}) > 0$ for all $k \ge 1$, $x^{(k)} \to x^*$ as $k \to \infty$, and the rate of convergence is quadratic.*

This result is a special case of the result for the multi-variable Newton's method which is proved in Fletcher (1980, p. 35). Unfortunately it is not easy to determine how close $x^{(1)}$ must be to x^*, so Newton's method is only really useful when you can find a good starting point. Note also that if x^* is a minimizer with $f''(x^*) = 0$ the the rate of convergence is only linear (for example $f(x) = x^4$ where $x^* = 0$, and Exercise 2).

4.3 Secant Method

A disadvantage of Newton's method is that it requires both first and second derivatives to be evaluated. The secant method approximates the second derivative $f''(x^{(k)})$ by a difference in first derivatives at the points $x^{(k)}$ and $x^{(k-1)}$, so

$$f''(x^{(k)}) \approx \frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Using this approximation to replace the second derivative term in Newton's method produces

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - f'(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} \\ &= \frac{f'(x^{(k)})x^{(k-1)} - f'(x^{(k-1)})x^{(k)}}{f'(x^{(k)}) - f'(x^{(k-1)})}. \end{aligned}$$

It can be shown [Lue73, page 141] that if $x^{(k)} \to x^*$ where $f'(x^*) = 0$ and $f''(x^*) > 0$ then the secant method has a superlinear rate of convergence (order of convergence ≈ 1.618).

Because derivative values at two points are required to approximate a second derivative the secant method requires two starting points, for example the points $x^{(1)}$ and $x^{(2)}$ in Table 4.3.1.

k	$x^{(k)}$	$f(x^{(k)})$	$f'(x^{(k)})$	approximation to $f''(x^{(k)})$
1	.5	-.22942554	.12241744	
2	.4	-.22941834	-.12106099	2.43478
3	.44972144	-.23246532	-.00112534	2.41215
4	.45017843	-.23246558	-.00001019	2.43492
5	.45018361	-.23246558	-.00000000	2.43517

Table 4.3.1: Secant method on Example 1

4.4 Polynomial interpolation

Frequently information (function values and possibly derivatives) at two or more points is used to find a low order polynomial which approximates the function. An interpolating polynomial has the same agrees exactly with the underlying function at the data points. This gives a system of equations which can be solved for the coefficients in the polynomial. The minimizer of the approximation polynomial is then used as an approximation to the minimizer of the function. The two most common cases are cubic interpolation when function values and derivatives are available at two points, and quadratic interpolation when only function values are available at three points. Algorithms and code, in either Fortran or C, are given in *Numerical Recipes* [PTVF92, Sections 10.2 and 10.3]. A classic reference for methods when only function values are available is Brent [Bre73].

4.4.1 Cubic Interpolation

If the function and derivative values at two points are available then the four pieces of information

$$x^{(k-1)}: \quad f(x^{(k-1)}) \quad f'(x^{(k-1)}) \quad \text{and} \quad x^{(k)}: \quad f(x^{(k)}) \quad f'(x^{(k)}).$$

can be used. A cubic model

$$p(x) = A\left(x - x^{(k-1)}\right)^3 + B\left(x - x^{(k-1)}\right)^2 + C\left(x - x^{(k-1)}\right) + D$$

interpolates this data. Let $\Delta_k = x^{(k)} - x^{(k-1)}$. The four parameters A, B, C, D are determined by the four equations

$$\begin{aligned}
f(x^{(k-1)}) &= p(x^{(k-1)}) = D \\
f'(x^{(k-1)}) &= p'(x^{(k-1)}) = C \\
f(x^{(k)}) &= p(x^{(k)}) = A\Delta_k^3 + B\Delta_k^2 + C\Delta_k + D \\
f'(x^{(k)}) &= p'(x^{(k)}) = 3A\Delta_k^2 + 2B\Delta_k + C.
\end{aligned}$$

If the points $x^{(k)}$ and $x^{(k-1)}$ are distinct then

$$\begin{aligned}
A &= \frac{-2\left(f(x^{(k)}) - f(x^{(k-1)})\right) + \left(f'(x^{(k-1)}) + f'(x^{(k)})\right)\Delta_k}{\Delta_k^3} \\
B &= \frac{3\left(f(x^{(k)}) - f(x^{(k-1)})\right) - \left(f'(x^{(k)}) + 2f'(x^{(k-1)})\right)\Delta_k}{\Delta_k^2}.
\end{aligned}$$

The stationary points of the cubic satisfy

$$p'(x) = 3A(x - x^{(k-1)})^2 + 2B(x - x^{(k-1)}) + C = 0,$$

so a local minimizer of the cubic model is

$$x_c^{(k)} = x^{(k-1)} + \frac{-B + \sqrt{B^2 - 3AC}}{3A}$$

as taking the positive square root ensures that $p''(x_c^{(k)}) > 0$.

Rounding errors can be amplified when two nearly equal quantities are subtracted (catastrophic cancellation). To avoid this the minimizer of the cubic should be calculated by

$$x_c^{(k)} = \begin{cases} x^{(k-1)} + \dfrac{-B + \sqrt{B^2 - 3AC}}{3A} & \text{if } B \leq 0, \\ x^{(k-1)} - \dfrac{C}{B + \sqrt{B^2 - 3AC}} & \text{if } B > 0. \end{cases} \tag{4.4.1}$$

The local minimizer of the cubic model can then be used as a new estimate of a minimizer of f, giving $x^{(k+1)} = x^{(k)}{}_c$. If $x^{(k)} \to x^*$ where $f'(x^*) = 0$ and $f''(x^*) > 0$ then cubic interpolation has a quadratic rate of convergence [Lue73, page 142], and will also find a minimizer of a cubic in one iteration.

4.4.2 Quadratic Interpolation

Frequently only function values and not first derivatives are available. Then a quadratic model $q(x) = Ax^2 + Bx + C$ can be fitted using function values at three different points. The three parameters A, B, and C are determined by the three equations

$$f(x^{(k)}) = q(x^{(k)}) = A(x^{(k)})^2 + Bx^{(k)} + C$$

$$f(x^{(k-1)}) = q(x^{(k-1)}) = A(x^{(k-1)})^2 + Bx^{(k-1)} + C$$

$$f(x^{(k-2)}) = q(x^{(k-2)}) = A(x^{(k-2)})^2 + Bx^{(k-2)} + C.$$

The minimizer of this quadratic model is $x_q^{(k)} = -B/2A$ if $q''(x_q^{(k)}) = A > 0$. Solving the equations for A and B gives

$$x_q^{(k)} = \frac{v_1 f(x^{(k-2)}) + v_2 f(x^{(k-1)}) + v_3 f(x^{(k)})}{2[u_1 f(x^{(k-2)}) + u_2 f(x^{(k-1)}) + u_3 f(x^{(k)})]}$$

provided

$$A = -\frac{u_1 f(x^{(k-2)}) + u_2 f(x^{(k)}) + u_3 f(x^{(k)})}{v_1 v_2 v_3} > 0,$$

where

$$u_1 = x^{(k)} - x^{(k-1)}, u_2 = x^{(k-2)} - x^{(k)}, u_3 = x^{(k-1)} - x^{(k-2)}$$

$$v_1 = x^{(k)^2} - x^{(k-1)^2}, v_2 = x^{(k-2)^2} - x^{(k)^2}, v_3 = x^{(k-1)^2} - x^{(k-2)^2}.$$

The minimizer of the quadratic model can then be used as a new estimate of a local minimizer of $f(x)$, giving $x^{(k+1)} = x_q^{(k)}$. If $x^{(k)} \to x^*$ where $f'(x^*) = 0$ and $f''(x^*) > 0$ then quadratic interpolation has a superlinear rate of convergence (order of convergence ≈ 1.3) [Lue73, page 143].

4.5 Bracketing Methods

A major disadvantage of Newton's method, the secant method, and quadratic interpolation is that the quadratic model on which they are based has a minimizer only if its second derivative is positive. Thus these methods are not globally convergent. It is highly desirable that a method works from any starting point(s), and that some guaranteed information about a minimizer can be provided.

Definition 4.5.1 *An interval* $[a, b]$ brackets *a minimizer if a local minimizer is guaranteed to lie in the open interval* (a, b).

A bracketing interval is also referred to as an interval of uncertainty, as just where a minimizer lies within (a, b) is uncertain. Some commonly used conditions that ensure that an interval $[a, b]$ brackets a minimizer are:

1. *A three point pattern:* If $a < x < b$, $f \in C[a,b]$ and

$$f(a) > f(x) \text{ and } f(b) > f(x) \tag{4.5.1}$$

then $[a,b]$ brackets a minimizer.

2. If $a < b$, $f \in C^1[a,b]$ and any of the conditions

$$\begin{aligned} &f'(a) < 0 \text{ and } f(b) \geq f(a), \\ &f'(b) > 0 \text{ and } f(a) \geq f(b), \\ &f'(a) < 0 \text{ and } f'(b) > 0, \end{aligned} \tag{4.5.2}$$

are satisfied then $[a,b]$ brackets a minimizer.

If a method starts with a bracketing interval and reduces the length of the bracketing interval on each iteration then the method must be globally convergent. Thus a practical algorithm usually finds a bracket on a minimum first, then uses cubic interpolation (or quadratic interpolation if derivatives are not available) to produce a new estimate $x^{(k+1)}$ of a local minimizer. A new bracket is then chosen from $x^{(k+1)}$ and the previous bracketing points.

The estimate $x^{(k+1)}$ is adjusted if it lies too close to one of the points already forming the bracket. Let $L^{(k)}$ denote the length of the k th bracketing interval. Then $x^{(k+1)}$ is chosen so that

$$|x^{(k+1)} - x| \geq \zeta L^{(k)} \text{ where } 0 < \zeta < 1$$

for all points x forming the current bracket. This ensures that $L^{(k+1)} \leq (1-\zeta)L^{(k)}$, so $L^{(k)} \leq (1-\zeta)^{k-1}L^{(1)}$, and hence $L^{(k)} \to 0$ as $k \to \infty$, ensuring global convergence of the method.

4.5.1 Bisection

Let f be a continuously differentiable function, and let $a^{(k)} < b^{(k)}$ be a bracket on a minimizer. Consider the midpoint

$$x_m^{(k)} = \frac{a^{(k)} + b^{(k)}}{2}.$$

If $a^{(k)}, x_m^{(k)}$ satisfies the bracketing conditions (4.5.2) then the new bracket is $[a^{(k+1)}, b^{(k+1)}] = [a^{(k)}, x_m^{(k)}]$. Otherwise $x_m^{(k)}, b^{(k)}$ satisfies the bracketing conditions (4.5.2) and the new bracket is $[a^{(k+1)}, b^{(k+1)}] = [x_m^{(k)}, b^{(k)}]$.

In both cases the length $L^{(k)} = b^{(k)} - a^{(k)}$ of the bracketing interval is halved, so $L^{(k+1)} = \frac{1}{2}L^{(k)}$. Hence $L^{(k+1)} = (\frac{1}{2})^k L^{(1)}$ so $L^{(k)} \to 0$ as $k \to \infty$ and the bisection will converge to a minimizer. The rate of convergence is linear, with rate constant $\beta = 0.5$.

4.5.2 Golden Section Search

The methods discussed so far assume that the function can be modelled by a quadratic or cubic polynomial. This is true if f is sufficiently smooth (differentiable). However sometimes the functions to be minimized functions which are only continuous and not continuously differentiable. The golden section method maintains a bracket on a minimum and reduces the length of this bracketing interval by a constant amount on each iteration using only one new function evaluation per iteration.

4.6 Nonsmooth Problems

Numerical methods based on quadratic or cubic interpolation, Newton and Second methods all require that the function to be minimized is at least twice continuously differentiable. Problems in which the objective function is continuous, but not continuously differentiable, are called *nonsmooth* problems. For nonsmooth problems points where the function has a discontinuity in its derivative can be a minimizer, as well as stationary points. If the function is not continuous then points where the function is discontinuous must also be considered.

Definition 4.6.1 (Critical points)
Consider the problem of minimizing $f(x)$ on $\Omega \subseteq \mathbb{R}$. The critical points of f on Ω are

1. *The end points of Ω.*
2. *Any points $x \in \Omega$ where $f(x)$ is not continuous,*
3. *Any points $x \in \Omega$ where $f'(x)$ is not continuous,*
4. *Any stationary points $x \in \Omega$ where $f'(x) = 0$.*

If $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz (i.e. for every $\delta > 0$ $|f(x) - f(y)| \leq |x - y|$ for all $|x - y \leq \delta$), then the (one-sided) directional derivative

$$f'(x; d) = \lim_{\alpha \to o^+} \frac{f(x + \alpha d) - f(x)}{\alpha} \tag{4.6.1}$$

exists. A convex function $f : \mathbb{R} \to \mathbb{R}$ is locally Lipschitz and hence continuous and directionally differentiable. Moreover for $n = 1$ only the directions $d = \pm 1$ are significant as the directional derivative is *positively homogeneous*

$$f'(x; \xi d) = \xi f'(x; d) \quad \forall \xi > 0.$$

Also

$$\begin{aligned} f'(x; +1) &= \lim_{\substack{x^{(k)} \downarrow x^* \\ f'(x^{(k)}) \ exists}} f'(x^{(k)}) \\ f'(x; -1) &= \lim_{\substack{x^{(k)} \uparrow x^* \\ f'(x^{(k)}) \ exists}} f'(x^{(k)}) \end{aligned}$$

Proposition 4.6.2 *Let $f : [a, b] \to \mathbb{R}$ be directionally differentiable.*

1. *If $x^* \in (a, b)$ and*
   (a) *$f'(x^*, -1) > 0$ and $f'(x^*, 1) > 0$ then x^* is a strict local minimizer of $f(x)$.*
   (b) *$f'(x^*, -1) < 0$ and $f'(x^*, 1) < 0$ then x^* is a strict local maximizer of $f(x)$.*
2. *If $x^* = a$ and*
   (a) *$f'(a, 1) > 0$ then x^* is a strict local minimizer of $f(x)$.*
   (b) *$f'(a, 1) < 0$ then x^* is a strict local maximizer of $f(x)$.*
3. *If $x^* = b$ and*
   (a) *$f'(b, -1) > 0$ then x^* is a strict local minimizer of $f(x)$.*
   (b) *$f'(b, -1) < 0$ then x^* is a strict local maximizer of $f(x)$.*

4.7 Exercises

1. Let $g(x) = f'(x)$.
   (a) Newton's method for solving $g(x) = 0$ uses the tangent line at $x^{(k)}$ to obtain a new estimate of a zero of $g(x)$. Show that this produces the same iteration formula as Newton's method for minimizing $f(x)$, except that there is no requirement that $f''(x^{(k)}) > 0$.
   (b) The Newton-Raphson method for solving $g(x) = 0$ uses the chord passing through the points $x^{(k)}, f(x^{(k)})$ and $x^{(k-1)}, f(x^{(k-1)})$ to obtain a new estimate of a zero of $g(x)$. Show that this produces the same iteration formula as the secant method for minimizing $f(x)$.

2. Consider minimizing $f(x) = (x - \xi)^p$, where $p \geq 2$.
   (a) Show that if the requirement that $f''(x^{(k)}) > 0$ is ignored then Newton's method reduces to
   $$x^{(k+1)} = \frac{(p-2)x^{(k)} + \xi}{(p-1)}.$$
   (b) Show that the iteration scheme of part a) is globally convergent to the point ξ for all p, even though this point is the minimizer of $f(x)$ only for even p.
   (c) Show that for $p > 2$ the rate of convergence is linear with rate $(p-2)/(p-1)$.
   (d) Calculate how many iterations Newton's method will require to satisfy the following convergence criteria when $x^{(1)} = \xi + 1$, $p = 10$ and $\epsilon = 10^{-7}$.
      i. $|x^{(k)} - \xi| < \epsilon$
      ii. $f(x^{(k)}) - f(\xi) < \epsilon$
      iii. $|f'(x^{(k)})| < \epsilon$
      iv. $|x^{(k+1)} - x^{(k)}| < \epsilon$

3. Discuss how you would expect the following methods to perform on the following problems.
   (a) Maximize $f(x) = e^{-(x-1)^2}$ using
      i. Newton's method starting at $x^{(1)} = 0$.
      ii. Cubic interpolation with initial bracket [0 , 10].
   (b) Minimize $f(x) = |x^2 - 2x - 3|$ on $[-4, 5]$ using
      i. Newton's method starting at $x^{(1)} = -2$.
      ii. The secant method starting at $x^{(1)} = -4$ and $x^{(2)} = -2$.
      iii. Quadratic interpolation.
      iv. Golden section search.
   (c) Use the MATLAB routine `fmin` to try to solve these two problems, and see if the results are what you expected.

4. Minimize $f(x) = (x-1)^4$
   (a) on $[0, 5]$ using Newton's method starting at $x^{(1)} = 5$.
   (b) on $[2, 5]$ using quadratic interpolation.

5. For the following one-dimensional problems find all the critical points, and identify them as (strict)local or (strict)global minimizers or maximizers or neither a minimizer nor a maximizer. Verfiy your answers by plotting the functions using `Maple` or MATLAB.
   (a) $f(x) = \dfrac{x}{1 - |x|}$ on $\Omega = [-4, 8]$.
   (b) $f(x) = |x^2 - 2x|$ on $\Omega = [-3, 4)$.
   (c) $f(x) = \begin{cases} e^{-x} & \text{if } x > 0; \\ |x+1| & \text{if } x \leq 0. \end{cases}$ on $\Omega = \mathbb{R}$.
   (d) $f(x) = \begin{cases} \ln(x^2 - 8x + 17) + 2 & \text{if } x > 1; \\ |x^3 + 2x^2 - 8x| & \text{if } x \leq 1. \end{cases}$ on $\Omega = (-\infty, 10]$.
   (e) $f(x) = \begin{cases} 5e^{x^2 - 10x + 24} & \text{if } x \geq 4; \\ |x^2 - 2x - 3| & \text{if } x < 4 \end{cases}$ on $\Omega = [-3, \infty)$.

6. **[H]**The linear discrete time system $y^{(k+1)} = Ay^{(k)}$, where $y^{(k)} \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times m}$ is *stable* if the spectral radius $\rho(A) < 1$ ($\rho(A)$ is the maximum modulus eigenvalue of A). Similarly a linear continuous time system governed by the differential equation $\frac{dy(t)}{dt} = Ay(t)$ is *stable* if all the eigenvalues of the matrix A have negative real part.

   (a) Find, if possible, the values of the parameter $x \in \Re$ which will make the discrete time system governed by the matrix
   $$A = \begin{bmatrix} 1+x & 1 \\ -x & 1+x \end{bmatrix}$$
   stable by minimizing
   $$f(x) = \rho(A(x)) = \max_{i=1,\ldots,m} |\lambda_i(A(x))|,$$
   where $\lambda_i(A(x))$ denotes the ith eigenvalue of $A(x)$. Note that A is not symmetric so the eigenvalues of A may be complex.

   (b) Find, if possible, the values of the parameter $x \in \Re$ which will make the continuous time system governed by the matrix
   $$A = \begin{bmatrix} -1+2x & 1 \\ x & -1+2x \end{bmatrix}$$
   stable by minimizing
   $$f(x) = \max_{i=1,\ldots,m} Re[\lambda_i(A(x))],$$
   where $Re[z]$ denotes the real part of the complex number z.

Chapter 5

Methods for Unconstrained Problems

5.1 Introduction

Although many real-life optimization problems involve constraints, the methods used to solved them are heavily based on numerical methods for unconstrained problems. This chapter considers numerical methods that are used to try to find a local solution $\mathbf{x}^*$ of

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \quad f(\mathbf{x}).$$

It is assumed the function $f(\mathbf{x})$ is at least once continuously differentiable, so a minimizer $\mathbf{x}^*$ of the unconstrained problem must be a *stationary point* with $\nabla f(\mathbf{x}^*) = 0$. Unless $f(\mathbf{x})$ is convex, or has some other special properties, there will be no guarantee that a global minimizer has been found.

Numerical algorithms require a starting point (initial guess) $\mathbf{x}^{(1)}$. They then typically try to generate a sequence of points $\{\mathbf{x}^{(k)}\}$ which progresses steadily towards a neighbourhood of a local minimizer $\mathbf{x}^*$, and then converges rapidly to the point $\mathbf{x}^*$ itself. As the problem is to find a local minimizer of $f(\mathbf{x})$ a natural condition to impose on the sequence of iterates $\{\mathbf{x}^{(k)}\}$ is that

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}). \tag{5.1.1}$$

A method which generates iterates satisfying (5.1.1) is a *descent method*. This ensures that each iteration make some improvement in the objective value, but is not enough to guarantee convergence to a local minimizer.

At a point $\mathbf{x}^{(k)}$ methods often determine a step or search direction $\mathbf{d}^{(k)}$ based on a local model of the objective function about the point $\mathbf{x}^{(k)}$. The new point may be $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$, or a step $\alpha^{(k)} > 0$ in the direction $\mathbf{d}^{(k)}$ so $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$. A direction $\mathbf{d}^{(k)}$ is a *descent direction* for $f(x)$ at $\mathbf{x}^{(k)}$ if

$$\mathbf{d}^{(k)^T} \nabla f(\mathbf{x}^{(k)}) < 0. \tag{5.1.2}$$

The following abbreviations are used: the objective value at the point $\mathbf{x}^{(k)}$ is $f^{(k)} \equiv f(\mathbf{x}^{(k)})$, the objective gradient at $\mathbf{x}^{(k)}$ is $\mathbf{g}^{(k)} \equiv \nabla f(\mathbf{x}^{(k)})$, and the objective Hessian at $\mathbf{x}^{(k)}$ is $G^{(k)} \equiv \nabla^2 f(\mathbf{x}^{(k)})$.

5.1.1 Quadratic models

Many minimization methods find a search direction $\mathbf{d}^{(k)}$ at $\mathbf{x}^{(k)}$ by minimizing a quadratic model of the objective $f(x)$ about $\mathbf{x}^{(k)}$. The direction $\mathbf{d}^{(k)}$ is a solution of

$$\underset{\mathbf{d} \in \mathbb{R}^n}{Minimize} \quad q_k(\mathbf{d}) = f^{(k)} + \mathbf{d}^T\mathbf{g}^{(k)} + \tfrac{1}{2}\mathbf{d}^T H^{(k)}\mathbf{d}. \tag{5.1.3}$$

Here $H^{(k)}$ can be the identity matrix I, the Hessian $G^{(k)}$ or a symmetric approximation to the Hessian.

A quadratic model is only appropriate if the objective function is twice continuously differentiable. It is the simplest unconstrained problem that has a unique minimizer. If $H^{(k)}$ is positive definite the minimizer $\mathbf{d}^{(k)}$ of (5.1.3) is the solution of the n by n symmetric system of linear equations

$$H^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}. \tag{5.1.4}$$

If $H^{(k)}$ is positive definite the minimizer $\mathbf{d}^{(k)}$ of the quadratic model (5.1.3) is a descent direction for $f(\mathbf{x})$ at $\mathbf{x}^{(k)}$ as

$$\mathbf{g}^{(k)^T}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)^T}H^{(k)^{-1}}\mathbf{g}^{(k)} < 0.$$

Example 5.1.1 (Quadratic models) *Consider the one-dimensional functions*

$$f(x) = -\cos(x) \text{ and } \hat{f}(x) = \max[-\cos(x),\ (2x-9)/10].$$

At the point $x^{(k)} = -0.6$ the value of the function is $f^{(k)} = -\cos(x^{(k)}) = -0.8253356149$ while $g^{(k)} = \sin(x^{(k)}) = -0.5646424734$ and $G^{(k)} = cos(x^{(k)}) = 0.8253356149$. The minimizer of the quadratic model is $d^{(k)} = -G^{(k)^{-1}}g^{(k)} = 0.6841368083$. The predicted minimizer of $f(x)$ is $x^{(k)} + d^{(k)} = 0.08413680834$, which is a good estimate of the minimizer $x^* = 0$. The function $f(x)$ and the quadratic model are plotted in the first plot in Figure 5.1.1 Making a quadratic model of $\hat{f}(x)$ at $x^{(k)}$ using the second order Taylor

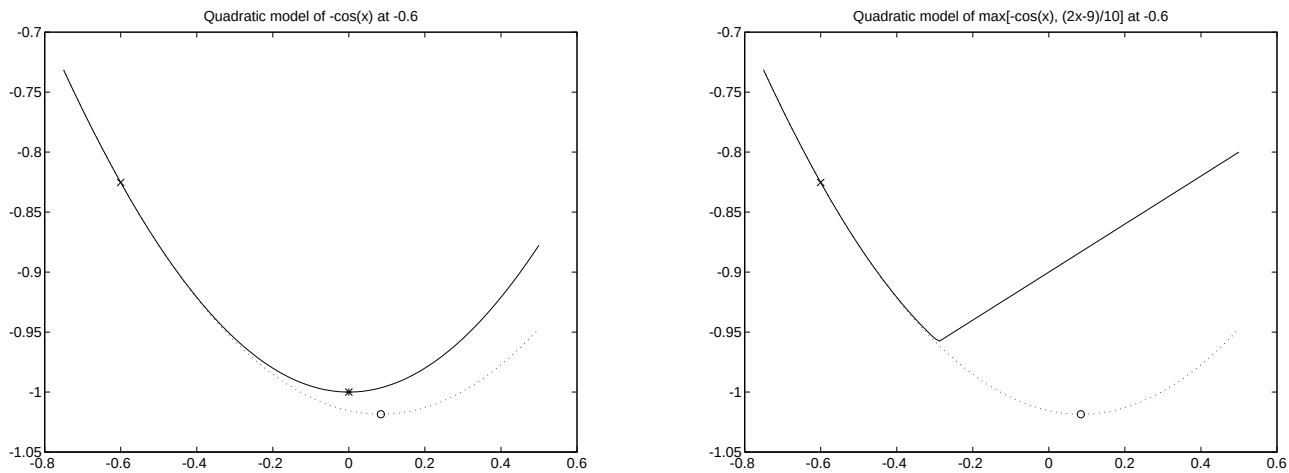


Figure 5.1.1: Quadratic models of $f(x) = -\cos(x)$ and $\hat{f}(\mathbf{x}) = \max[-\cos(x),\ (2x-9)/10]$

series produces the same estimate of the minimizer. However from the graph of $\hat{f}(x)$ in the second plot in Figure 5.1.1 the minimizer occurs when both component functions achieve the max. This implies

$$-cos(x) = (2x-9)/10 \Longrightarrow \hat{x}^* = -0.2905016017 \text{ and } \hat{f}(\hat{x}^*) = -0.9581003205.$$

The max function $\hat{f}(x)$, which is not continuously differentiable at $\hat{x}^*$, is adequately modelled by a quadratic function only *in a small neighbourhood of $x^{(k)}$. Thus the estimate $x^{(k)} + d^{(k)} \approx 0.08$ produced by the quadratic model is not a good estimate of $\hat{x}^* \approx -0.29$.*

5.1.2 Line search methods

Given a search direction $\mathbf{d}^{(k)}$ at a point $\mathbf{x}^{(k)}$ the next iterate is $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$ where $\alpha^{(k)}$ is either an exact or approximate minimizer of $\ell_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$. If $\mathbf{d}^{(k)}$ is a descent direction then there exists an $\bar{\alpha}^{(k)} > 0$ such that

$$f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}) \quad \forall\, \alpha \in (0, \bar{\alpha}^{(k)}].$$

The step $\alpha^{(k)}$ is chosen by exactly or approximately minimizing the function value

$$\ell_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) \tag{5.1.5}$$

along the line $\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}$. An exact minimizer satisfies

$$\ell_k'(\alpha) = \nabla f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})^T \mathbf{d}^{(k)} = 0. \tag{5.1.6}$$

Approximate line search conditions are discussed in Section 5.1.3. If $\mathbf{d}^{(k)}$ is a descent direction for f at $\mathbf{x}^{(k)}$ then a line search produces a step $\alpha^{(k)} > 0$ such that $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.

Example 5.1.2 (Exact line search for quadratic functions) *Let $f(\mathbf{x})$ be a strictly convex quadratic function*

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G \mathbf{x} + \mathbf{g}_0^T \mathbf{x} + f_0.$$

Equation (5.1.6) for an exact minimizer implies

$$\Big(G(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) + \mathbf{g}_0\Big)^T \mathbf{d}^{(k)} = 0 \Longrightarrow \alpha^{(k)} = -\frac{(G\mathbf{x}^{(k)} + \mathbf{g}_0)^T \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}} = -\frac{\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)}}{\mathbf{d}^{(k)T} G \mathbf{d}^{(k)}}. \tag{5.1.7}$$

As $f(\mathbf{x})$ is strictly convex, G is positive definite so $\ell_k''(\alpha) = \mathbf{d}^{(k)T} G \mathbf{d}^{(k)} > 0$ for any $\mathbf{d}^{(k)} \neq 0$. Hence $\alpha^{(k)}$ is a minimizer of $\ell_k(\alpha)$. If $\mathbf{d}^{(k)}$ is a descent direction, so $\nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} < 0$, then $\alpha^{(k)} > 0$.

5.1.3 Approximate line searches

When the current iterate $\mathbf{x}^{(k)}$ is far from a solution it may not be efficient to do the work required to find an exact minimizer of the objective along the line $\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}$. The overall efficiency of the solution method may be improved by finding an approximate minimizer $\alpha^{(k)}$ of

$$\ell_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

The new iterate is $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)}$ and a new search direction $\mathbf{d}^{(k+1)}$ can be calculated incorporating information gained in moving from $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k+1)}$.

If $\mathbf{d}^{(k)}$ is a descent direction ($\ell_k'(0) < 0$) then attention can be restricted to $\alpha \geq 0$. To prove convergence it is important that the objective value is reduced by at least a factor that depends on the size of the step taken. The accuracy of the line search must also be controlled. An exact line search must satisfy $\ell_k'(\alpha^*) = 0$. Thus an approximate minimizer of $\ell_k(\alpha)$ is any value of α which satisfies

$$\ell_k(\alpha) \leq \ell_k(0) + \alpha \rho \ell_k'(0) \tag{5.1.8}$$

$$|\ell_k'(\alpha)| \leq -\sigma \ell_k'(0) \tag{5.1.9}$$

where $0 < \rho < \sigma < 1$.

The inequalities on ρ and σ guarantee that for a continuously differentiable objective $f(\mathbf{x})$ and a descent direction $\mathbf{d}^{(k)}$ both (5.1.8) and (5.1.9) can be satisfied. Let $[0, \alpha_f]$ be an interval satisfying (5.1.8), and let $[\alpha_{sl}, \alpha_{su}]$ be an interval satisfying (5.1.9). Then, as $\sigma > 0$, $\alpha^* \in [\alpha_{sl}, \alpha_{su}]$. The set of approximate minimizers is the non-empty set

$$[0, \alpha_f] \cap [\alpha_{sl}, \alpha_{su}].$$

In the limit $\sigma \to 0^+$ the slope condition (5.1.9) implies $\ell_k'(\alpha) = 0$, which is a necessary condition for a local minimizer. Thus a small values of σ (e.g. 0.01) corresponds to a strong (accurate) line search, while a large value of σ (e.g. 0.9) corresponds to a weak (not very accurate) line search.

Example 5.1.3 (Approximate line search) *Consider the function*

$$f(\mathbf{x}) = 3x_1^4 - 4x_1^3 - 12x_1^2 + (x_2 - 1)^2 + 12.$$

Let

$$\mathbf{x}^{(k)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{d}^{(k)} = \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix} \Longrightarrow \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} = \begin{bmatrix} 1 + \frac{\alpha}{2} \\ 0 \end{bmatrix}.$$

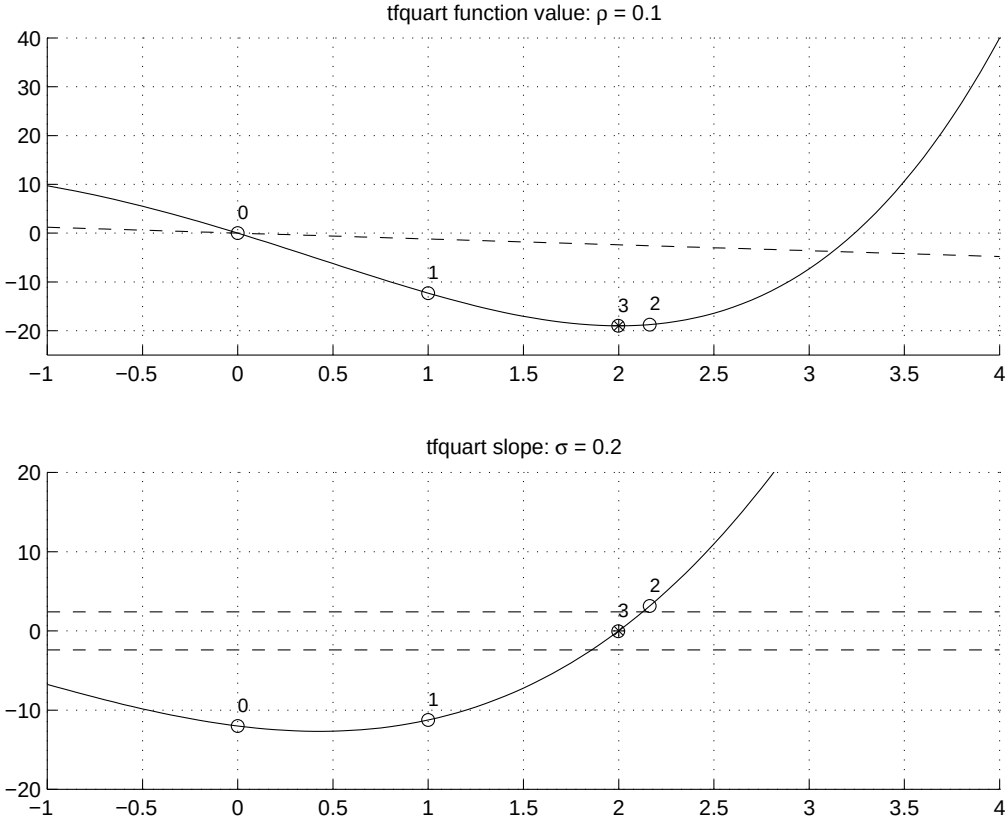


Figure 5.1.2: Approximate line search with stronger slope restriction

Then

$$\begin{aligned}
\ell_k(\alpha) &= 3(1+\frac{\alpha}{2})^4 - 4(1+\frac{\alpha}{2})^3 - 12(1+\frac{\alpha}{2})^2 + 13 \\
&= \frac{3}{16}\alpha^4 + \alpha^3 - \frac{3}{2}\alpha, \\
\ell'(\alpha) &= \frac{3}{4}\alpha^3 + 3\alpha^2 - 3\alpha - 12.
\end{aligned}$$

The exact minimizer must satisfy $\ell_k'(\alpha) = 0$*, which has solutions* $\alpha = -4, -2, 2$*. As* $\mathbf{d}^{(k)}$ *is a descent direction the desired minimizer must be positive, so* $\alpha^* = 2$*. Substitution shows that* $\ell_k(\alpha^*) = -19$ *and* $\ell_k''(\alpha^*) = 18 > 0$*, so* $\alpha^* = 2$ *is at least a local minimizer of* $\ell_k(\alpha)$*.*

As $\ell_k(0) = f(\mathbf{x}^{(k)}) = 0$ *and* $\ell_k'(0) = \nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)} = -12$ *the approximate line search conditions (5.1.8) and (5.1.9) become*

$$\frac{3}{16}\alpha^4 + \alpha^3 - \frac{3}{2}\alpha \le -12\alpha\rho \tag{5.1.10}$$

$$|\frac{3}{4}\alpha^3 + 3\alpha^2 - 3\alpha - 12| \le 12\sigma \tag{5.1.11}$$

A `Maple` *script for this example is in the file* `lsmap.txt`*. The points* α *satisfying (5.1.10) and (5.1.11) are illustrated for* $\rho = 0.1$ *and* $\sigma = 0.2$ *in Figure 5.1.2, and for* $\rho = 0.4$ *and* $\sigma = 0.6$ *in Figure 5.1.3. For* $\sigma = 0.2$ *the set of points satisfying the approximate line search conditions is a smaller interval around the exact minimizer than for* $\sigma = 0.6$*.*

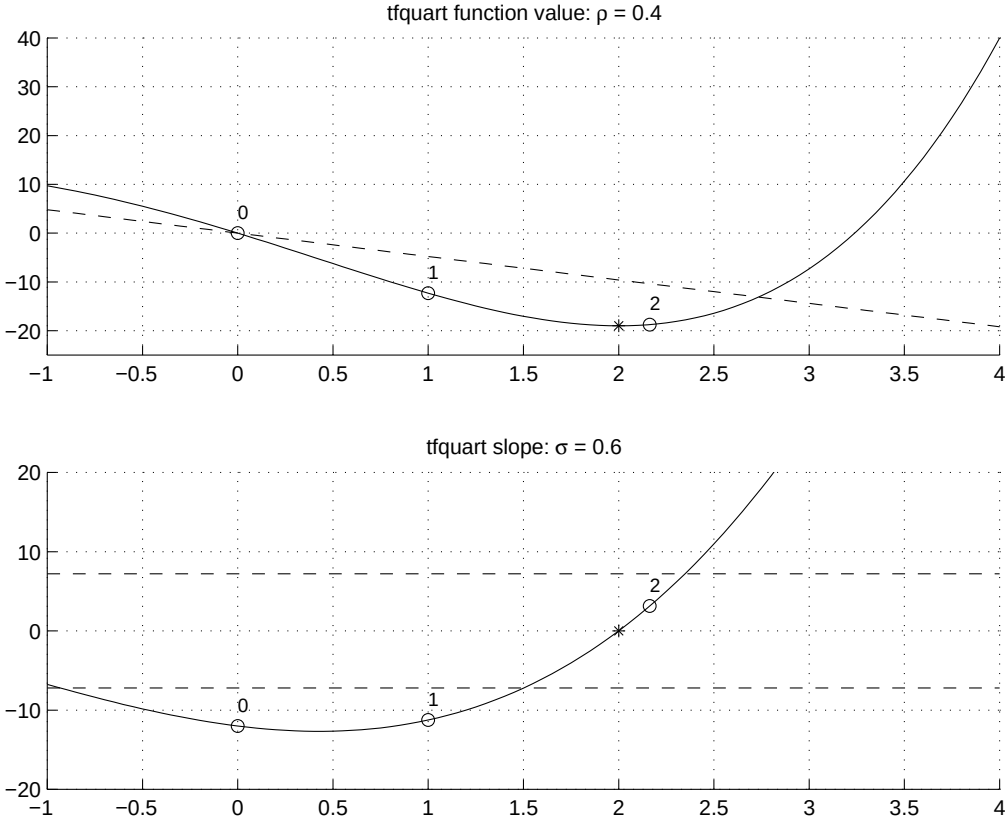


Figure 5.1.3: Approximate line search with weaker slope restriction

One simple method of finding a point $\alpha^{(k)}$ such that $f(\mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)})$ is to use an *Armijo Line search* [Arm66]. Let $0 < \tau < 1$. An Armijo line search uses $\alpha^{(k)} = \tau^{j^{(k)}}$ where $j^{(k)}$ is the smallest integer $j \geq 0$ satisfying

$$f(\mathbf{x}^{(k)} + \tau^j \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}) + \tau^j \rho \mathbf{d}^{(k)^T} \mathbf{g}^{(k)}. \tag{5.1.12}$$

Although very simple this is not as efficient as using a line search based on the one-dimensional quadratic or cubic approximation methods (see Chapter 4). Approximate line search algorithms with some interesting examples are discussed by Moré and Theunte [MT94]. Fletcher [Fle87] gives some of the convergence results that can be proved using approximate line searches.

5.1.4 Trust region methods

Algorithms often calculate a step or search direction $\mathbf{d}^{(k)}$ based on a local quadratic model of the objective function at an iterate $\mathbf{x}^{(k)}$. Local models, for example those obtained from a second order Taylor series expansion, may only be reasonable approximations in some neighbourhood of $\mathbf{x}^{(k)}$. Trust region methods explicitly restrict the new point $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$ to a neighbourhood of the current point $\mathbf{x}^{(k)}$ by restricting the size of the step $\mathbf{d}^{(k)}$. Thus $\mathbf{d}^{(k)}$ solves

$$\begin{array}{ll} \mathit{Minimize} & q_k(\mathbf{d}) = f^{(k)} + \mathbf{d}^T\mathbf{g}^{(k)} + \frac{1}{2}\mathbf{d}^T H^{(k)}\mathbf{d} \\ \mathbf{d} \in \mathbb{R}^n & \\ \mathit{Subject\ to} & \|\mathbf{d}\| \leq \delta^{(k)}. \end{array} \tag{5.1.13}$$

The trust region radius $\delta^{(k)}$ is updated based on the agreement between the function decrease predicted by the quadratic model and the actual function decrease in f when moving from $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k)} + \mathbf{d}^{(k)}$. A

line search is not used.

The trust region constraint $\|\mathbf{d}\| \leq \delta^{(k)}$ ensures that the feasible region is closed and bounded and the quadratic model $q_k(\mathbf{d})$ is a continuous function, so global extrema exist for problem (5.1.13). As any norm is a convex function, the feasible region $\{\mathbf{d} \in \mathbb{R}^n : \|\mathbf{d}\| \leq \delta^{(k)}\}$ is a convex set. Also if $H^{(k)}$ is positive definite then the quadratic model is a strictly convex function. Thus when $H^{(k)}$ is positive definite the unique local minimizer is the global minimizer. However when $H^{(k)}$ is indefinite there many be many different local minimizer. For $\delta^{(k)} > 0$ any feasible point is a regular point, so the local minimizer must be a constrained stationary point. The global minimizer will be the constrained stationary point with the lowest objective value.

In unconstrained optimization the norm defining the trust region is usually the 2-norm, so

$$\|\mathbf{d}\|_2 \leq \delta^{(k)} \iff \mathbf{d}^T\mathbf{d} \leq {\delta^{(k)}}^2.$$

Using the 2-norm gives a circular trust region in $\mathbb{R}^2$, as illustrated in Figure 5.1.4. A circular trust region

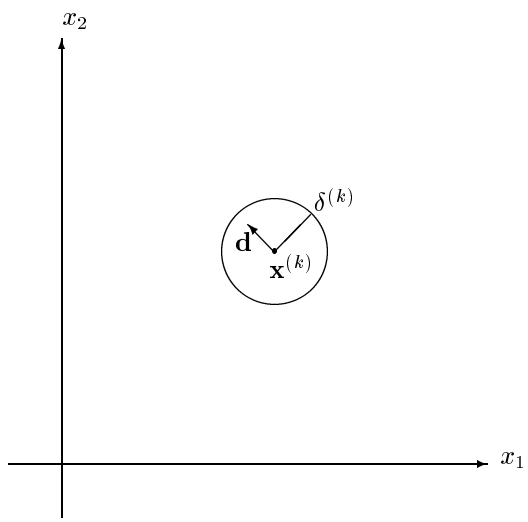


Figure 5.1.4: 2-norm trust region

assumes the variables are similarly scaled. An elliptic trust region, but still one with its axes parallel to the $\mathbf{x}$ coordinates is defined by the constraint

$$\|\mathbf{x}\|_D \leq \delta^{(k)} \iff \mathbf{x}^T D\mathbf{x} \leq {\delta^{(k)}}^2,$$

where D is a diagonal matrix with positive diagonal elements. More generally if A is any n by n symmetric positive definite matrix then the constraint

$$\|\mathbf{x}\|_A \leq \delta^{(k)} \iff \mathbf{x}^T A\mathbf{x} \leq {\delta^{(k)}}^2,$$

defines an elliptic trust region with axes parallel to the eigenvectors of A. The sensitivity to the scaling of the variables is a disadvantage of trust region methods.

The radius $\delta^{(k)}$ of the trust region is updated based on the agreement between the quadratic model $q_k(\mathbf{d}^{(k)})$ and the objective function value $f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)})$. Let

$$\begin{aligned} \Delta f^{(k)} &= f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}) \\ \Delta \mathbf{q}^{(k)} &= f(\mathbf{x}^{(k)}) - q_k(\mathbf{d}^{(k)}). \end{aligned}$$

The actual change in the objective value when a step of $\mathbf{d}^{(k)}$ is taken from $\mathbf{x}^{(k)}$ is $\Delta f^{(k)}$, while $\Delta \mathbf{q}^{(k)}$ is the predicted change in the objective value based on the quadratic model q_k. For any $\delta^{(k)} > 0$ the point

$\mathbf{d} = 0$ is feasible and $q_k(0) = f^{(k)}$. If the global minimizer $\mathbf{d}^{(k)}$ of (5.1.13) is non-zero then $\Delta\mathbf{q}^{(k)} > 0$. Then

$$r^{(k)} = \frac{\Delta f^{(k)}}{\Delta q^{(k)}}$$

is a measure of the accuracy with which the quadratic model $q_k(\mathbf{d})$ approximates $f(\mathbf{x}^{(k)} + \mathbf{d})$ over the current trust region. A value $r^{(k)} = 1$ indicates very good agreement. If $\mathbf{d}^{(k)} \neq 0$ is the global minimizer of (5.1.13) then $q_k(\mathbf{d}^{(k)}) < q_k(0) = f^{(k)}$, so $\Delta q_k > 0$. Thus a value $r^{(k)} < 0$ indicates $f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}) > f^{(k)}$, so that the agreement is so poor that the step $\mathbf{d}^{(k)}$ increases, rather than decreases, the objective function. A typical scheme for updating the trust region radius is

Algorithm 5.1.4 (Trust region algorithm)
Let $0 < \rho_0 < \rho_d < 0.5 < \rho_i < 1$, $0 < \tau_d < 1$ and $1 < \tau_i$. At an iterate $\mathbf{x}^{(k)}$ let $\mathbf{d}^{(k)}$ be a global solution of (5.1.13).

1. *Only update the point if the function value decreases:*
   If $r^{(k)} < \rho_0$ then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$; otherwise $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$.
2. (a) *If the agreement is poor decrease the trust region radius:*
   If $r^{(k)} < \rho_d$ then $\delta^{(k+1)} = \tau_d \delta^{(k)}$.
   (b) *If the agreement if good and the trust region is active increase the trust region radius:*
   If $r^{(k)} > \rho_i$ and $\|\mathbf{d}^{(k)}\| = \delta^{(k)}$ then $\delta^{(k+1)} = \tau_i \delta^{(k)}$.
   (c) *Otherwise keep the same trust region radius: $\delta^{(k+1)} = \delta^{(k)}$.*

A trust region algorithm is not very sensitive to the values of the parameters $\rho_0, \rho_d, \rho_i, \tau_d, \tau_i$, with typical values being $\rho_0 = 0.01$, $\rho_d = 0.25$, $\rho_i = 0.75$, $\tau_d = 0.25$ and $\tau_i = 2$.

5.1.5 Convergence conditions

Iterative methods for minimizing a function produce a sequence of points $\{\mathbf{x}^{(k)}\}$ which converges to a local minimizer $\mathbf{x}^*$ of $f(\mathbf{x})$. As $f(\mathbf{x})$ is assumed to be continuous $f(\mathbf{x}^{(k)}) \to f^* \equiv f(\mathbf{x}^*)$ as $k \to \infty$. Ideally the iteration would be stopped when either or both of

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \epsilon_{\mathbf{x}} \tag{5.1.14a}$$

$$f^{(k)} - f^* \leq \epsilon_f \tag{5.1.14b}$$

are satisfied. Here $\epsilon_{\mathbf{x}} > 0$ and $\epsilon_f > 0$ are small positive tolerances reflecting the desired accuracy in the solution point and the objective value respectively. The norm in (5.1.14a) could be either the 2-norm or the ∞-norm. For example using the ∞-norm and $\epsilon_{\mathbf{x}} = \frac{1}{2} \times 10^{-5}$ would guarantee that each component has an *absolute error* less than 5×10^{-6}. If the components of $\mathbf{x}^*$ are large or the magnitude of the minimum objective value f^* is large then it would be more appropriate to use a test based on the *relative error*

$$\frac{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} \leq \epsilon_{\mathbf{x}} \tag{5.1.15a}$$

$$\frac{f^{(k)} - f^*}{|f^*|} \leq \epsilon_f. \tag{5.1.15b}$$

Unfortunately the conditions (5.1.14a) – (5.1.15b) are not practical as $\mathbf{x}^*$ and f^* are generally not known.

If $f^{(k)} \to f^*$ then $f^{(k)} - f^{(k+1)} \to 0^+$ as a descent method has $f^{(k+1)} \leq f^{(k)}$. The converse does not hold, as it is possible for $f^{(k)} - f^{(k+1)} \to 0$ without $f^{(k)} \to f^*$. Similarly if $\mathbf{x}^{(k)} \to \mathbf{x}^*$ then $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \to 0$, but $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \to 0$ does not necessarily imply $\mathbf{x}^{(k)} \to \mathbf{x}^*$. As $\mathbf{x}^*$ is not known before hand one or more of the convergence conditions

$$f^{(k)} - f^{(k+1)} \leq \epsilon_f \tag{5.1.16a}$$

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\| \leq \epsilon_{\mathbf{x}} \tag{5.1.16b}$$

$$\|\mathbf{g}^{(k)}\| \leq \epsilon_{\mathbf{g}} \tag{5.1.16c}$$

are typically used. In (5.1.16b) either the 2-norm or ∞-norm may be used. Condition (5.1.16c) is only appropriate for unconstrained problems.

5.1.6 Desirable properties

1. A method has *global convergence* if it converges to a stationary point from *any* starting point $\mathbf{x}^{(1)}$.

   It is always useful to use as good a starting point as possible. This may be obtained from the physical problem or from the previous solution of a similar problem. A bad starting point may also introduce numerical problems, for example undefined functions ($\log(x)$ for $x < 0$), underflow (e^{-x^2} for large x) or overflow (e^{x^2} for large x).

2. A method has *quadratic termination* if it finds the minimizer of a strictly convex quadratic function of n variables in at most n iterations.

   This means the method exhibits finite termination on a positive definite quadratic function.

3. The *order of convergence* is the largest ν such that

$$\lim_{k\to\infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^\nu} < \infty.$$

   Examples of the different rates of convergence was given in Section 1.5.2. Faster methods exhibit quadratic (order $\nu = 2$) or superlinear (order $\nu > 1$) rates of convergence. The rate of convergence is an *asymptotic* property describing the convergence of the iterates when they are very close to a solution. It does not say anything about the efficiency of the method when it is far from a minimizer.

4. A method has *affine invariance* if the affine transformation $\mathbf{y} = A\mathbf{x} + b$ does not change the iterates.

5.1.7 Exercises

1. Let $f(\mathbf{x}) = x_1^2 + 10x_2^2$, $\mathbf{x}^{(1)} = [\ 1 \quad 0.1\ ]^T$ and $\mathbf{d}^{(1)} = [\ -2 \quad -2\ ]^T$.

   (a) Find an expression for $\ell_1(\alpha) = f(\mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)})$, the value of the function f along the line passing through $\mathbf{x}^{(1)}$ in the direction $\mathbf{d}^{(1)}$.

   (b) Verify that

$$\ell'(\alpha) \equiv \frac{d\,\ell_1(\alpha)}{d\alpha} = \nabla f(\mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)})^T\mathbf{d}^{(1)}.$$

   (c) Show that $\mathbf{d}^{(1)}$ is a descent direction for f at $\mathbf{x}^{(1)}$.

   (d) Find the exact minimizer $\alpha^{(1)}$ of $\ell_1(\alpha)$.

   (e) Show that $\mathbf{d}^{(1)}$ is tangential to the contour of f passing through the point $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha^{(1)}\mathbf{d}^{(1)}$.

   (f) Find all the points α which satisfy the approximate line search conditions

$$\begin{aligned} \ell_1(\alpha) &\le \ell(0) + \alpha\rho\ell_1'(0) \\ |\ell_1'(\alpha)| &\le -\sigma\ell_1'(0) \end{aligned}$$

   with $\rho = 0.1$ and

   i. $\sigma = 0.2$.

   ii. $\sigma = 0.9$.

2. At the point $\mathbf{x}^{(k)} = [\ 2 \quad 2\ ]^T$ sketch the following trust regions. All have trust region radius $\delta^{(k)} = 1$.

   (a) $\|\mathbf{x} - \mathbf{x}^{(k)}\|_2 \le \delta^{(k)}$.

   (b) $\|\mathbf{x} - \mathbf{x}^{(k)}\|_D \le \delta^{(k)}$ where $D = \mathrm{diag}(\frac{1}{2}, 3)$.

   (c) $\|\mathbf{x} - \mathbf{x}^{(k)}\|_A \le \delta^{(k)}$ where $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$.

   (d) $\|\mathbf{x} - \mathbf{x}^{(k)}\|_\infty \le \delta^{(k)}$.

3. Let $f(\mathbf{x}) = (x_1 - 1)^6 + 100x_2^2$ and $\mathbf{x}^{(k)} = [\ 1.01 \quad 0\ ]^T$. Show that $f^{(k)} - f^* = 10^{-12}$ and $\|\mathbf{g}^{(k)}\| = 6 \times 10^{-10}$, but that $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0.01$. Hence it is possible for the convergence criteria based on the objective function and the gradient to be satisfied but there still to be a significant error in $\mathbf{x}^{(k)}$.

5.2 Steepest descent

A basic property of the gradient is that at a point $\bar{\mathbf{x}}$ the direction $\mathbf{d} = -\nabla f(\bar{\mathbf{x}})$ is the direction in which the function decreases most rapidly (see Example 3.2.13). Similarly $\nabla f(\bar{\mathbf{x}})$ is the direction in which the function increases most rapidly at $\bar{\mathbf{x}}$. This is a local property in that the steepest descent direction depends on the point $\bar{\mathbf{x}}$. Using the steepest descent direction in conjunction with a line search produces the steepest descent method.

Algorithm 5.2.1 (Steepest Descent)

1. **Initialisation:** *Given a starting point* $\mathbf{x}^{(1)}$*, a tolerance* $\epsilon_g > 0$*, and a maximum number of iterations* k_{max}*, set* $k = 1$*.*
2. *Calculate the function value* $f^{(k)} = f(\mathbf{x}^{(k)})$ *and the gradient* $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$*.*
3. *While* $k < k_{max}$ *and* $\|\mathbf{g}^{(k)}\| > \epsilon_g$
   (a) **Search direction:** *Calculate the steepest descent search direction* $\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$*.*
   (b) **Line search:** *Calculate an (exact or approximate) minimizer* $\alpha^{(k)}$ *of* $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$ *to find*
      i. **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$*.*
      ii. **New function value and gradient:** $f^{(k+1)} = f(\mathbf{x}^{(k+1)})$ *and* $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$*.*
   (c) *Increase iteration counter:* $k = k + 1$*.*

   End while loop.

The steepest descent method requires the evaluation of the function value $f(\mathbf{x}^{(k)})$ and the gradient $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$. Hence the line search can be based on a cubic model of the objective function (see Section 4.4). The line search also provides the new point $\mathbf{x}^{(k+1)}$ and the function value $f(\mathbf{x}^{(k+1)})$ and the gradient $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$.

The attraction of the steepest descent method are its simplicity and nice global convergence properties. Fletcher [Fle87] proves that, under mild conditions on $f(\mathbf{x})$, the steepest descent method (with exact or approximate line searches) is guaranteed to converge to a stationary point. As only function values and gradient are evaluated this is the strongest result that can be expected. The primary disadvantage of the steepest descent method is its slow rate of convergence. Even on a strictly convex quadratic function with positive definite Hessian the order of convergence is only linear. Define the spectral condition number

$$\kappa(G) = \frac{\lambda_{max}(G)}{\lambda_{min}(G)},$$

where $\lambda_{max}(G)$ is the largest eigenvalue of G and $\lambda_{min}(G)$ is the smallest eigenvalue of G. Luenberger [Lue73] proves that the rate of convergence is linear with

$$\beta \leq \frac{\kappa(G) - 1}{\kappa(G) + 1}. \tag{5.2.1}$$

If the Hessian is a scalar multiple of the identity matrix, so $G = \eta I$, then $\lambda_{max}(\eta I) = \lambda_{min}(\eta I) = \eta$. In this case $\kappa(G) = 1$ and $\beta \leq 0$, indicating the rate of convergence is superlinear or better. However if the condition number of G is large, then the bound (5.2.1) can become arbitrarily close to 1. For example $\kappa(G) = 100$ implies $\beta \leq \frac{99}{101} \approx 0.98$. The actual rate of convergence depends on the starting point $\mathbf{x}^{(1)}$. If $\mathbf{x}^{(1)} - \mathbf{x}^*$ is a scalar multiple of an eigenvector of G then the steepest descent method converges in one iteration (see Question 2 in Exercises 5.2.1 and Example 5.2.2)

Example 5.2.2 (Steepest Descent on a quadratic function) *Consider the function* $f(x) = f_0 + \mathbf{g}_0^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T G\mathbf{x}$ *with* $n = 2$ *and*

$$f_0 = 1, \quad \mathbf{g}_0 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}, \quad G = \begin{bmatrix} 5 & -2 \\ -2 & 1 \end{bmatrix}. \tag{5.2.2}$$

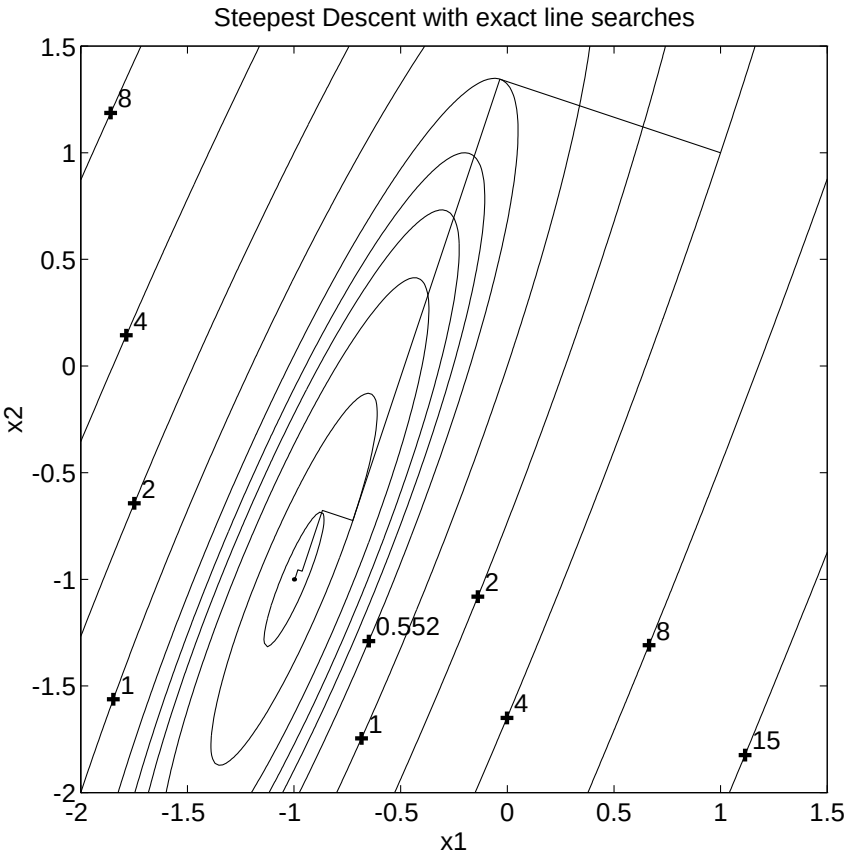


Figure 5.2.1: Steepest descent starting from $\mathbf{x}^{(1)} = [1\ 1]^T$

As G is positive definite the exact minimizer of this strictly convex quadratic function is the solution of $G\mathbf{x} + \mathbf{g}_0 = 0$, which gives

$$\mathbf{x}^* = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \text{ where } f^* = 0.$$

A summary of the iterates produced using the steepest descent method with an exact line search (5.1.7) are given in Tables 5.2.1 and 5.2.2. The algorithm used convergence criteria of $\|\mathbf{g}^{(k)}\|_2 < 10^{-12}$ and $\|\mathbf{x} - \mathbf{x}^{(k-1)}\|_2 < 10^{-10}$, which were not satisfied. The iteration terminated because for iteration $k = 20$ the line search has an initial slope of $\mathbf{d}^{(k)^T}\mathbf{g}^{(k)} = -6.21 \times 10^{-17}$. As this is less than the double precision relative rounding error of 2.2×10^{-16}, the direction $\mathbf{d}^{(k)}$ is not regarded as a descent direction. A convergence test like $f^{(k-1)} - f^{(k)} < 10^{-14}$ would have been satisfied. Figure 5.2.1 gives the contours of the quadratic function defined by (5.2.2) and plots the iterates starting from $\mathbf{x}^{(1)} = [\,1 \quad 1\,]^T$.

Table 5.2.2 lists the iterates and the errors produced by the steepest descent method with exact line searches, starting from $\mathbf{x}^{(1)} = [\,1 \quad 1\,]^T$.

Table 5.2.3 lists the norms of the errors using the 2-norm $\|\mathbf{e}\|_2^2 = \mathbf{e}^T\mathbf{e}$ and using the G-norm $\|\mathbf{e}\|_G^2 = \mathbf{e}^T G\mathbf{e}$ arising from the positive definite Hessian G. For linear convergence $\|\mathbf{e}^{(k+1)}\|/\|\mathbf{e}^{(k)}\| \to \beta$ while $\|\mathbf{e}^{(k+1)}\|/\|\mathbf{e}^{(k)}\| \to \infty$, which is clearly the case in Table 5.2.3. The logs of the errors are plotted in Figure 5.2.1. The errors using the 2-norm converge unevenly, while errors measured using the G-norm lie exactly on the line. Using either norm produces a value of $\beta = 0.37$.

The eigenvalues of G are 5.82843 and 0.1715731, so the spectral condition number of G is $\kappa(G) =$

k	n_f	$f^{(k)}$	$\|\mathbf{g}^{(k)}\|$	$\|\mathbf{d}^{(k)}\|$	${\mathbf{d}^{(k)}}^T\mathbf{g}^{(k)}$	$\alpha^{(k)}$	$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|$
1	1	4.00000e+00	6.32e+00	6.32e+00	-4.00e+01	0.172	
2	2	5.51724e-01	4.36e-01	4.36e-01	-1.90e-01	5.000	1.09e+00
3	3	7.60999e-02	8.72e-01	8.72e-01	-7.61e-01	0.172	2.18e+00
4	4	1.04965e-02	6.02e-02	6.02e-02	-3.62e-03	5.000	1.50e-01
5	5	1.44780e-03	1.20e-01	1.20e-01	-1.45e-02	0.172	3.01e-01
6	6	1.99696e-04	8.30e-03	8.30e-03	-6.89e-05	5.000	2.07e-02
7	7	2.75443e-05	1.66e-02	1.66e-02	-2.75e-04	0.172	4.15e-02
8	8	3.79922e-06	1.14e-03	1.14e-03	-1.31e-06	5.000	2.86e-03
9	9	5.24030e-07	2.29e-03	2.29e-03	-5.24e-06	0.172	5.72e-03
10	10	7.22800e-08	1.58e-04	1.58e-04	-2.49e-08	5.000	3.95e-04
11	11	9.96965e-09	3.16e-04	3.16e-04	-9.97e-08	0.172	7.89e-04
12	12	1.37512e-09	2.18e-05	2.18e-05	-4.74e-10	5.000	5.44e-05
13	13	1.89672e-10	4.36e-05	4.36e-05	-1.90e-09	0.172	1.09e-04
14	14	2.61620e-11	3.00e-06	3.00e-06	-9.02e-12	5.000	7.51e-06
15	15	3.60867e-12	6.01e-06	6.01e-06	-3.61e-11	0.172	1.50e-05
16	16	4.97935e-13	4.14e-07	4.14e-07	-1.72e-13	5.000	1.04e-06
17	17	6.86118e-14	8.29e-07	8.29e-07	-6.87e-13	0.172	2.07e-06
18	18	9.54792e-15	5.71e-08	5.71e-08	-3.27e-15	5.000	1.43e-07
19	19	1.22125e-15	1.14e-07	1.14e-07	-1.31e-14	0.172	2.86e-07
20	19	3.33067e-16	7.88e-09	7.88e-09	-6.21e-17		1.97e-08

Table 5.2.1: Steepest descent on quadratic function

k	$x_1^{(k)}$	$x_2^{(k)}$	$e_1^{(k)}$	$e_2^{(k)}$
1	1.0000000000	1.0000000000	2.00e+00	2.00e+00
2	-0.0344827586	1.3448275862	9.66e-01	2.34e+00
3	-0.7241379310	-0.7241379310	2.76e-01	2.76e-01
4	-0.8668252081	-0.6765755054	1.33e-01	3.23e-01
5	-0.9619500595	-0.9619500595	3.80e-02	3.80e-02
6	-0.9816310632	-0.9553897249	1.84e-02	4.46e-02
7	-0.9947517323	-0.9947517323	5.25e-03	5.25e-03
8	-0.9974663535	-0.9938468586	2.53e-03	6.15e-03
9	-0.9992761010	-0.9992761010	7.24e-04	7.24e-04
10	-0.9996505315	-0.9991512908	3.49e-04	8.49e-04
11	-0.9999001519	-0.9999001519	9.98e-05	9.98e-05
12	-0.9999517975	-0.9998829367	4.82e-05	1.17e-04
13	-0.9999862278	-0.9999862278	1.38e-05	1.38e-05
14	-0.9999933514	-0.9999838533	6.65e-06	1.61e-05
15	-0.9999981004	-0.9999981004	1.90e-06	1.90e-06
16	-0.9999990829	-0.9999977729	9.17e-07	2.23e-06
17	-0.9999997380	-0.9999997380	2.62e-07	2.62e-07
18	-0.9999998735	-0.9999996928	1.26e-07	3.07e-07
19	-0.9999999639	-0.9999999639	3.61e-08	3.61e-08
20	-0.9999999826	-0.9999999576	1.74e-08	4.24e-08

Table 5.2.2: Steepest descent iterates $\mathbf{x}^{(k)}$ and errors $\mathbf{e}^{(k)}$

k	$\|\mathbf{e}^{(k)}\|_2$	$\frac{\|\mathbf{e}^{(k)}\|_2}{\|\mathbf{e}^{(k-1)}\|_2}$	$\frac{\|\mathbf{e}^{(k)}\|_2}{\|\mathbf{e}^{(k-1)}\|_2^2}$	$\|\mathbf{e}^{(k)}\|_G$	$\frac{\|\mathbf{e}^{(k)}\|_G}{\|\mathbf{e}^{(k-1)}\|_G}$	$\frac{\|\mathbf{e}^{(k)}\|_G}{\|\mathbf{e}^{(k-1)}\|_G^2}$
2	2.5e+00	9.0e-01	3.2e-01	1.1e+00	3.7e-01	1.3e-01
3	3.9e-01	1.5e-01	6.1e-02	3.9e-01	3.7e-01	3.5e-01
4	3.5e-01	9.0e-01	2.3e+00	1.4e-01	3.7e-01	9.5e-01
5	5.4e-02	1.5e-01	4.4e-01	5.4e-02	3.7e-01	2.6e+00
6	4.8e-02	9.0e-01	1.7e+01	2.0e-02	3.7e-01	6.9e+00
7	7.4e-03	1.5e-01	3.2e+00	7.4e-03	3.7e-01	1.9e+01
8	6.7e-03	9.0e-01	1.2e+02	2.8e-03	3.7e-01	5.0e+01
9	1.0e-03	1.5e-01	2.3e+01	1.0e-03	3.7e-01	1.3e+02
10	9.2e-04	9.0e-01	8.8e+02	3.8e-04	3.7e-01	3.6e+02
11	1.4e-04	1.5e-01	1.7e+02	1.4e-04	3.7e-01	9.8e+02
12	1.3e-04	9.0e-01	6.3e+03	5.2e-05	3.7e-01	2.6e+03
13	1.9e-05	1.5e-01	1.2e+03	1.9e-05	3.7e-01	7.1e+03
14	1.7e-05	9.0e-01	4.6e+04	7.2e-06	3.7e-01	1.9e+04
15	2.7e-06	1.5e-01	8.8e+03	2.7e-06	3.7e-01	5.1e+04
16	2.4e-06	9.0e-01	3.3e+05	1.0e-06	3.7e-01	1.4e+05
17	3.7e-07	1.5e-01	6.4e+04	3.7e-07	3.7e-01	3.7e+05
18	3.3e-07	9.0e-01	2.4e+06	1.4e-07	3.7e-01	1.0e+06
19	5.1e-08	1.5e-01	4.6e+05	5.1e-08	3.7e-01	2.7e+06
20	4.6e-08	9.0e-01	1.8e+07	1.9e-08	3.7e-01	7.3e+06

Table 5.2.3: Estimating rates of convergence

$5.82843/0.1715731 \approx 34$. *Thus the rate β of linear convergence satisfies*

$$\beta \leq \frac{\kappa(G)-1}{\kappa(G)+1} = \frac{33}{35} = 0.943.$$

The observed linear rate of convergence when starting from $\mathbf{x}^{(1)} = [\ 1 \quad 1\ ]^T$ of 0.37 is well under the bound of 0.943.

The choice of starting point has a major impact on the actual rate of convergence produced from the algorithm. For example if $\mathbf{x}^{(1)} = [\ 0 \quad \sqrt{2}\ ]^T$, so $\mathbf{x}^{(1)} - \mathbf{x}^*$ is a multiple of an eigenvector of G, then the steepest descent method with exact line searches converges in one iteration. However starting points can also be chosen to get very close to the worst case. The performance of the steepest descent starting from

$$\mathbf{x}^{(1)} = \begin{bmatrix} 0.1 \\ \sqrt{2} \end{bmatrix}$$

is given in Figure 5.2.3. The iteration was stopped at a maximum number of iterations of $k = 101$ with $\mathbf{x}^{(k)} = [\ -0.99740 \quad -0.99530\ ]^T$ and $f^{(k)} = 3.5021288094 \times 10^{-6}$, $\|\mathbf{g}^{(k)}\|_2 = 1.66e \times 10^{-3}$ and $\|\mathbf{e}^{(k)}\| = 6.3 \times 10^{-3}$. The observed rate of linear convergence is $\beta \approx 0.941$.

The bound (5.2.1) on the rate of linear convergence can be used to estimate the number of iterations required to achieve a particular accuracy $\epsilon_{\mathbf{x}}$. Using the steepest descent method with exact line searches on a convex quadratic function with Hessian G, then

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|} \leq \frac{\kappa(G)-1}{\kappa(G)+1}.$$

Suppose that $\|\mathbf{x}^{(1)} - \mathbf{x}^*\| \leq \eta$. Then

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \left(\frac{\kappa-1}{\kappa+1}\right)^k \|\mathbf{x}^{(1)} - \mathbf{x}^*\| \leq \left(\frac{\kappa-1}{\kappa+1}\right)^k \eta.$$

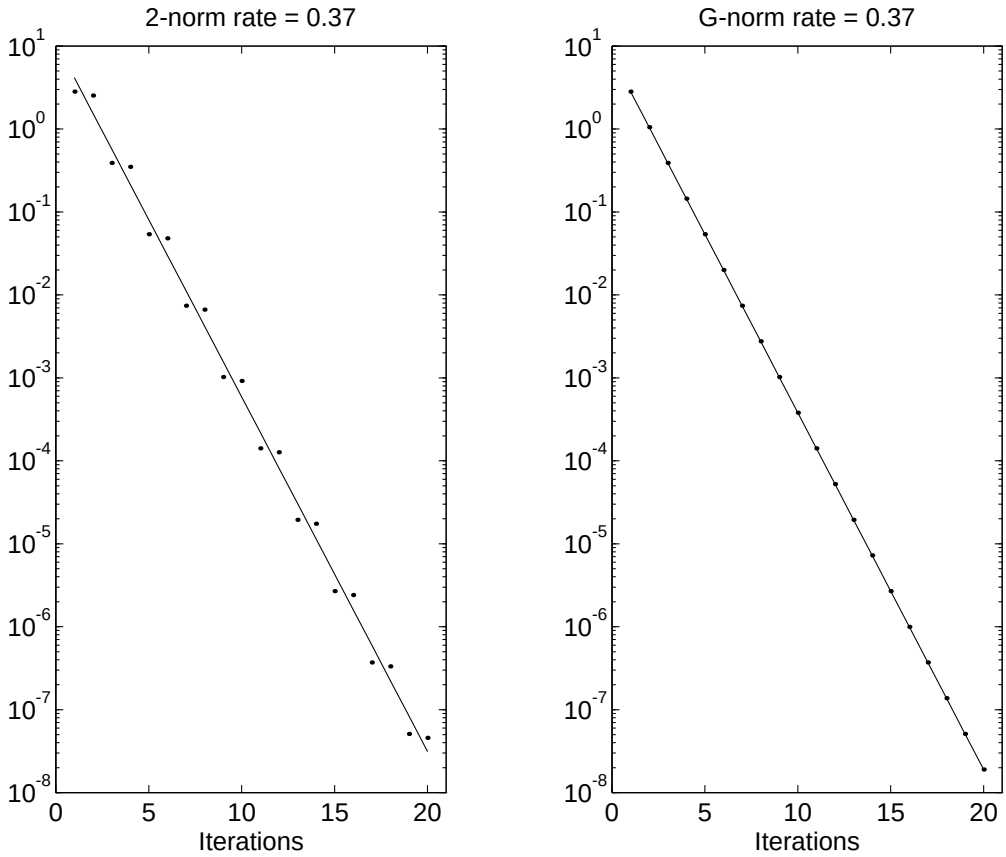


Figure 5.2.2: Steepest descent starting from $\mathbf{x}^{(1)} = [1\ 1]^T$

Choosing k as the smallest integer that satisfies

$$\left(\frac{\kappa - 1}{\kappa + 1}\right)^k \eta \leq \epsilon_{\mathbf{x}} \tag{5.2.3}$$

guarantees that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \epsilon_{\mathbf{x}}.$$

Then taking logs of (5.2.3) gives k as the smallest integer such that

$$k \geq \log\left(\frac{\epsilon_{\mathbf{x}}}{\eta}\right) / \log\left(\frac{\kappa - 1}{\kappa + 1}\right). \tag{5.2.4}$$

5.2.1 Exercises

1. Consider the function $f(\mathbf{x}) = x_1^2 - 4x_1 + 5x_2^2 + 30x_2 + 50$ and the starting point $\mathbf{x}^{(1)} = [1\ \ -2]^T$.
   (a) Show that f is a strictly convex quadratic function and find its minimizer $\mathbf{x}^*$ and the corresponding minimum f^*.
   (b) Carry out one iteration (i.e. find $\mathbf{x}^{(2)}$ and $f^{(2)}$) of the steepest descent method with exact line searches starting from $\mathbf{x}^{(1)}$.
   (c) Find the condition number κ of the Hessian G of f using the 2-norm. Hence estimate the largest number of iterations that the steepest descent method, starting from $\mathbf{x}^{(1)}$ and using exact line searches, will take to find the minimizer of f with absolute error less than 10^{-5}.

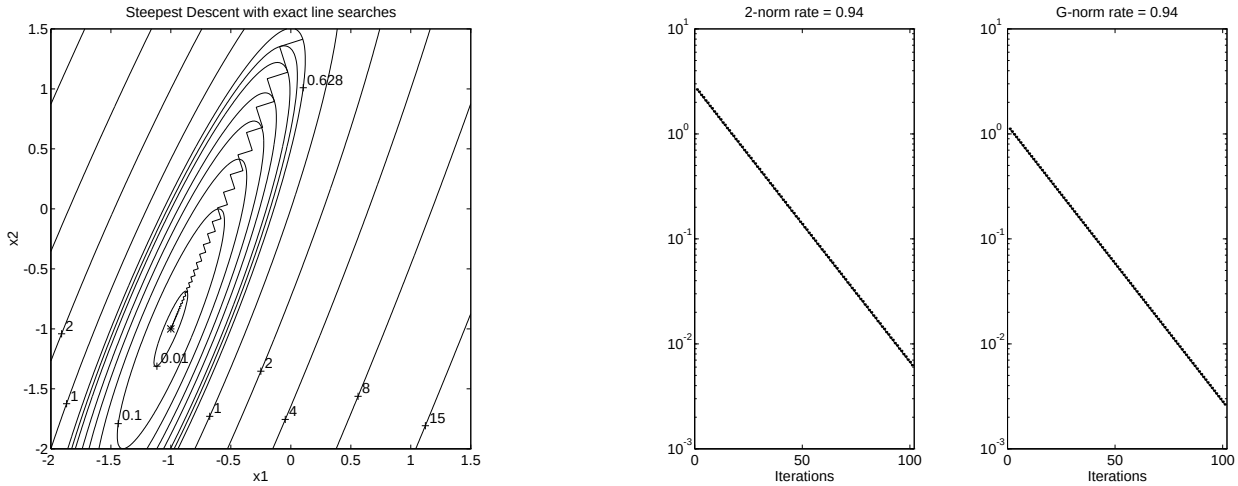


Figure 5.2.3: Steepest descent starting from $[0.1\ \sqrt{2}]$

2. Show that if $f(\mathbf{x})$ is a strictly convex quadratic function then

   (a) if $\mathbf{x}^{(1)} - \mathbf{x}^*$ is an eigenvector of G then the steepest descent method with exact line searches will converge in one iteration.

   (b) if the condition number of $\nabla^2 f(\mathbf{x})$ is one, then the steepest descent method with exact line searches will converge in 1 iteration from any starting point.

5.3 Newton's method

Newton's method is based in the quadratic model of the objective function $f(\mathbf{x})$ obtained from a second order Taylor series expansion. At the iterate $\mathbf{x}^{(k)}$

$$f(\mathbf{x}^{(k)} + \mathbf{d}) \approx q_k(\mathbf{d}) \equiv f^{(k)} + \mathbf{d}^T \mathbf{g}^{(k)} + \tfrac{1}{2}\mathbf{d}^T G^{(k)} \mathbf{d}.$$

The quadratic function $q_k(d)$ models the function $f(\mathbf{x})$ in some neighbourhood of the point $\mathbf{x}^{(k)}$. Newton's method chooses the step as the minimizer of $q_k(\mathbf{d})$, so $\mathbf{d}^{(k)}$ solves

$$\nabla q_k(\mathbf{d}) = G^{(k)}\mathbf{d} + \mathbf{g}^{(k)} = 0. \tag{5.3.1}$$

If $\nabla^2 q_k(\mathbf{d}) = G^{(k)}$ is positive definite then $\mathbf{d}^{(k)}$ is the minimizer of $q_k(\mathbf{d})$.

The basic Newton method always uses a step of 1, so $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$. It may be modified to include an exact or approximate line search, so $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$. With a line search it the step $\alpha^{(k)} = 1$ is typically tried first, to see if it satisfies the approximate line search conditions (5.1.8) and (5.1.9). This is because the quadratic model provides an estimate of the step to the minimum along the line $\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}$, not just a search direction as with the steepest descent direction. If $G^{(k)}$ is positive definite and $\mathbf{x}^{(k)}$ is not a stationary point ($\mathbf{g}^{(k)} \neq 0$) then the Newton direction $\mathbf{d}^{(k)} = -{G^{(k)}}^{-1}\mathbf{g}^{(k)}$ is a descent direction as

$$\mathbf{d}^{(k)^T}\mathbf{g}^{(k)} = -\mathbf{g}^{(k)^T}{G^{(k)}}^{-1}\mathbf{g}^{(k)} < 0.$$

Algorithm 5.3.1 (Newton's Method)

1. **Initialization:** *Given a starting point* $\mathbf{x}^{(1)}$, *a tolerances for the convergence conditions and a maximum number of iterations* k_{max}, *set* $k = 1$.

2. *Calculate the gradient* $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ *and the Hessian* $G^{(k)} = \nabla^2 f(\mathbf{x}^{(k)})$.

3. *While $k < k_{max}$ and convergence conditions not satisfied*

   (a) **Search direction:** *Calculate the Newton direction by solving $G^{(k)}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$.*

   (b) **Line search:** *Calculate an (exact or approximate) minimizer $\alpha^{(k)}$ of $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$, starting with $\alpha^{(k)} = 1$, to find*

      i. **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$.

      ii. **New function value, gradient and Hessian:** $f^{(k+1)} = f(\mathbf{x}^{(k+1)})$, $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$, $G^{(k+1)} = \nabla^2 f(\mathbf{x}^{(k)})$.

   (c) *Increase iteration counter: $k = k + 1$.*

   End while loop.

The search direction $\mathbf{d}^{(k)}$ is obtained by solving $G^{(k)}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$. If $G^{(k)}$ is singular then the search direction is not well-defined. If $G^{(k)}$ is nonsingular, but not positive definite, the symmetric linear system $G^{(k)}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$ can be solved for $\mathbf{d}^{(k)}$, but there is no guarantee that the resulting direction $\mathbf{d}^{(k)}$ is a descent direction. When $G^{(k)}$ is positive definite the linear system (5.3.1) can be efficiently solved using a Cholesky factorization $G^{(k)} = L^{(k)}{L^{(k)}}^T$ where $L^{(k)}$ is lower triangular. The Cholesky factorization can be written as $G^{(k)} = L^{(k)}D^{(k)}{L^{(k)}}^T$ where $L^{(k)}$ is unit lower triangular.

Example 5.3.2 (LDL^T factorization)

$$G^{(k)} = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 5 & -1 \\ 0 & -1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ 0 & -\frac{1}{4} & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & \frac{11}{4} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{4} \\ 0 & 0 & 1 \end{bmatrix}$$

When $G^{(k)}$ may be indefinite the Bunch-Kaufman factorization

$$G^{(k)} = L^{(k)}D^{(k)}{L^{(k)}}^T \tag{5.3.2}$$

where $L^{(k)}$ is unit lower triangular, and $D^{(k)}$ is block diagonal with 1 by 1 or 2 by 2 blocks on its diagonal is required (see Question 3 and Section A.7.2). A symmetric matrix $G^{(k)}$ is positive definite if and only if $D^{(k)}$ in (5.3.2) is diagonal with positive diagonal elements.

A major attraction of Newton's method is its fast rate of convergence. A proof of Proposition 5.3.3 can be found in Fletcher [Fle87, Theorem ??].

Proposition 5.3.3 *Let $f \in C^3(\mathbb{R}^n)$ and let $\mathbf{x}^*$ be a local minimizer of f where $G^* \equiv \nabla^2 f(\mathbf{x}^*)$ is positive definite. If $\mathbf{x}^{(1)}$ is sufficient close to $\mathbf{x}^*$ then Newton's method with a step $\alpha^{(k)} \equiv 1$ for all k, is well defined (i.e. $G^{(k)}$ is positive definite) and the iterates $\{\mathbf{x}^{(k)}\}$ converge to $\mathbf{x}^*$ as a second order rate.*

If $\mathbf{x}^*$ is a local minimizer where G^* is singular then if Newton's method converges the rate of convergence is typically only linear. Unfortunately Proposition 5.3.3 does not give any idea just how close $\mathbf{x}^{(1)}$ must be to $\mathbf{x}^*$ for the proposition to apply. In many physical situations a good starting point is known (or can be found). Proposition (5.3.3) gives no indication how Newton's method will behave if the starting point $\mathbf{x}^{(1)}$ is far from a minimizer. As $G^{(k)}$ may be singular away from a local minimizer the Newton direction may not be defined. Hence a global convergence result cannot be proved for Newton's method, either in its basic form with $\alpha^{(k)} \equiv 1$, or with an exact or approximate line search.

Example 5.3.4 (Newton's method on a quadratic function)

Minimize $f(\mathbf{x}) = x_1^2 + 10x_2^2$ starting from $\mathbf{x}^{(1)} = [\,1 \quad \frac{1}{10}\,]^T$ using Newton's method.

The gradient and Hessian of f are

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 20x_2 \end{bmatrix} \qquad G(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 20 \end{bmatrix}.$$

As $G(\mathbf{x})$ is independent of $\mathbf{x}$ and positive definite, f is a strictly convex quadratic function. Thus

$$\mathbf{g}^{(1)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

The basic Newton step is the solution of $G\mathbf{d}^{(1)} = -\mathbf{g}^{(1)}$ *giving*

$$\mathbf{d}^{(1)} = \begin{bmatrix} -1 \\ -\frac{1}{10} \end{bmatrix}.$$

Hence

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{d}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

where $f^{(2)} = 0$ *and* $\mathbf{g}^{(2)} = \mathbf{0}$. *This is the exact minimizer, which Newton's method found in 1 iteration as* f *is a strictly convex quadratic function.*

5.3.1 Modified Newton's method

A key problem with Newton's method is that the search direction is not defined if $G^{(k)}$ is singular. Even if $G^{(k)}$ is nonsingular, solving $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$ may not produce a minimizer of the quadratic model $q_k(\mathbf{d})$, and $\mathbf{d}^{(k)}$ may not be a descent direction. None of these problems arise if $G^{(k)}$ is positive definite.

One modification of Newton's method, often referred to as the Levenberg-Marquadt method, is to add a scalar multiple $\nu^{(k)}$ of the identity matrix I to the Hessian $G^{(k)}$ to get a positive definite matrix. The search direction $\mathbf{d}^{(k)}$ is then the solution of the linear system

$$(G^{(k)} + \nu^{(k)} I)\mathbf{d} = -\mathbf{g}^{(k)},$$

where $\nu \geq 0$ is chosen so that $G^{(k)} + \nu^{(k)} I$ is positive definite. Then $\mathbf{d}^{(k)}$ is a descent direction since

$$\mathbf{g}^{(k)^T}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)^T}\left(G^{(k)} + \nu^{(k)} I\right)^{-1}\mathbf{g}^{(k)} < 0$$

as long as $\mathbf{x}^{(k)}$ is not a stationary point ($\mathbf{g}^{(k)} \neq 0$).

If $\mathbf{x}^{(k)} \to \mathbf{x}^*$ where G^* is positive definite, then the method should take $\nu^{(k)} = 0$ for all $k \geq \bar{k}$. In this case the modified Newton method will exhibit both global convergence and a second order rate of convergence.

The remaining issue is how to choose $\nu^{(k)} \geq 0$ so that $G^{(k)} + \nu^{(k)} I$ is positive definite. Gill and Murray show how this can be done using the Bunch-Kaufman factorization of $G^{(k)}$.

Another, more fundamental, method for choosing $\nu^{(k)}$ is to use a trust region. As discussed in Section 5.1.4 the quadratic model $q_k(\mathbf{d})$ is only a local model of $f(\mathbf{x}^{(k)} + \mathbf{d})$, so is only likely to give a good approximation to f is a neighbourhood of $\mathbf{x}^{(k)}$. This suggests solving the subproblem

$$\min_{\|\mathbf{d}\| \leq \delta^{(k)}} q_k(\mathbf{d}), \tag{5.3.3}$$

where the constraint $\|\mathbf{d}\| \leq \delta^{(k)}$ represents a trust region about the current iterate $\mathbf{x}^{(k)}$. If $\mathbf{d}^{(k)}$ is the *global* minimizer of $q_k(\mathbf{d})$ over the set $\|\mathbf{d}\| \leq \delta^{(k)}$, then $q_k(\mathbf{d}^{(k)}) \leq q_k(0) = f^{(k)}$ as $\mathbf{d} = 0$ is always feasible. If $G^{(k)}$ is positive definite then (5.3.3) is a convex programming problem, so a stationary point is the global minimizer. If $G^{(k)}$ is not positive definite then it may be more difficult to find the global minimizer of (5.3.3).

The 2-norm gives a circular trust region about the iterate $\mathbf{x}^{(k)}$. This assumes that the variables are approximately equally scaled. If this is not the case an elliptic trust region with axes parallel to the coordinate axes can be specified by $\|\mathbf{d}\|_D \leq \delta^{(k)}$ where D is a positive definite diagonal matrix. More generally the axes of the trust region may be rotated by using $\|\mathbf{d}\|_A \leq \delta^{(k)}$ where A is any positive definite matrix. Using either of these norms requires a method for choosing the matrix D or A, which may need to change as the iterates $\mathbf{x}^{(k)}$ change. Another possibility is to use a trust region based on the infinity norm, namely $\|\mathbf{d}\|_\infty \leq \delta^{(k)}$. This corresponds to a box like trust region around the current point (see Figure A.6.1).

Using these norm to specify the trust region introduces problems when the trust region constraint needs to be differentiated. Different, but equivalent, algebraic descriptions of the trust regions are useful. For instance

- $\|\mathbf{d}\|_2 \leq \delta^{(k)} \iff \mathbf{d}^T\mathbf{d} \leq {\delta^{(k)}}^2$.
- $\|\mathbf{d}\|_A \leq \delta^{(k)} \iff \mathbf{d}^T A\mathbf{d} \leq {\delta^{(k)}}^2$.
- $\|\mathbf{d}\|_\infty \leq \delta^{(k)} \iff -\delta^{(k)} \leq d_i \leq \delta^{(k)}, i = 1, \ldots, n$.

The infinity norm trust region can be represented by simple lower and upper bounds on the variables, which is useful if a linear programming or quadratic programming subproblem is being solved. In unconstrained optimization the 2-norm trust region is most commonly used.

The trust region radius $\delta^{(k)}$ is updated using the agreement between the quadratic model and the objective function, as in Algorithm 5.1.4. The key question is how to efficiently solve the trust region subproblem (5.3.3). Consider a trust region using the 2-norm, so (5.3.3) is

$$\begin{array}{ll} \textit{Minimize} & q_k(\mathbf{d}) = f^{(k)} + \mathbf{d}^T\mathbf{g}^{(k)} + \frac{1}{2}\mathbf{d}^T G^{(k)}\mathbf{d} \\ \mathbf{d} \in \mathbb{R}^n & \\ \textit{Subject to} & c_1(\mathbf{d}) = \mathbf{d}^T\mathbf{d} - {\delta^{(k)}}^2 \leq 0. \end{array} \tag{5.3.4}$$

The two cases to be considered are when the constraint is not active, so $c_1(\mathbf{d}^{(k)}) < 0$, and when the constraint is active, so $c_1(\mathbf{d}) = 0$. If the constraint is not active the minimizer must be a stationary point of $q_k(\mathbf{d})$, so

$$\nabla q_k(\mathbf{d}) = 0 \Longrightarrow \mathbf{g}^{(k)} + G^{(k)}\mathbf{d} = 0.$$

The solution to $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$ will be the minimizer if $c_1(\mathbf{d}^{(k)}) \leq 0$ and $G^{(k)}$ is positive definite. If the trust region constraint is active then

$$\nabla q_k(\mathbf{d}) + \lambda_1 \nabla c_1(\mathbf{d}) = 0 \quad \Longrightarrow \quad G^{(k)}\mathbf{d} + \mathbf{g}^{(k)} + 2\lambda_1\mathbf{d} = 0 \tag{5.3.5}$$

$$c_1(\mathbf{d}) = 0 \quad \Longrightarrow \quad \mathbf{d}^T\mathbf{d} = {\delta^{(k)}}^2. \tag{5.3.6}$$

Equation (5.3.5) can be written as

$$\left(G^{(k)} + 2\lambda_1 I\right)\mathbf{d} = -\mathbf{g}^{(k)}.$$

If $G^{(k)} + 2\lambda_1 I$ is nonsingular then

$$\mathbf{d}^{(k)} = -\left(G^{(k)} + 2\lambda_1 I\right)^{-1}\mathbf{g}^{(k)}$$

where $\lambda_1 \geq 0$ satisfies

$$\psi(\lambda_1) \equiv {\mathbf{g}^{(k)}}^T \left(G^{(k)} + 2\lambda_1 I\right)^{-T} \left(G^{(k)} + 2\lambda_1 I\right)^{-1} \mathbf{g}^{(k)} = {\delta^{(k)}}^2. \tag{5.3.7}$$

The function $\psi(\lambda_1)$ may have several non-negative roots. As the global minimizer exists, it corresponds to the non-negative root of $\psi(\lambda_1)$ giving a value of $\mathbf{d}^{(k)}$ corresponding to the lowest function value of $q_k(\mathbf{d}^{(k)})$.

Trust region methods provide one of the two main ways, along with line search methods, of producing algorithms with global convergence. If it can be shown that the trust region radius is not active after a certain iteration, that is there exists a $\bar{k}$ such that $\|\mathbf{d}^{(k)}\| < \delta^{(k)}$ for all $k \geq \bar{k}$, the the method has the quadratic convergence of the underlying Newton method. The trust region methods described in this section are mainly used when the Hessian $\nabla^2 f(\mathbf{x})$ can be calculated, but is not necessarily positive definite.

5.3.2 Exercises

1. Consider the function $f(\mathbf{x}) = x_1^2 - 4x_1 + 5x_2^2 + 30x_2 + 50$ and the starting point $\mathbf{x}^{(1)} = [\,1 \quad -2\,]^T$. Use Newton's method to find the minimizer of f starting from $\mathbf{x}^{(1)}$. Comment on your results.

2. Consider the function $f(\mathbf{x}) = x_1^4 + x_1x_2 + (1 + x_2)^2$.

(a) i. Show that $f(\mathbf{x})$ has exactly one stationary point $\mathbf{x}^*$ which satisfies $8x_1^3 - x_1 - 2 = 0$ and $x_2 = -1 - \frac{x_1}{2}$, where $\frac{1}{2} < x_1 < 1$.

ii. Prove that this stationary point is a strict local minimizer of f.

(b) i. Find a local minimizer of f as accurately as your calculator allows using Newton's method starting from $\mathbf{x}^{(1)} = [\ \frac{3}{4} \quad -\frac{11}{8}\ ]^T$, with step sizes $\alpha^{(k)} = 1$ for all k.

ii. `Maple` can be used to show that

$$\mathbf{x}^* = \begin{bmatrix} \xi \\ -1 - \xi/2 \end{bmatrix} \quad \text{where} \quad \xi = \frac{\zeta^{\frac{2}{3}} + 6}{12\zeta^{\frac{1}{3}}}, \text{ and } \zeta = \left(216 + 6\sqrt{1290}\right).$$

Hence $\mathbf{x}^* \approx [\ 0.69588438611776391927 \quad -1.3479421930588819596\ ]^T$.
Estimate the order of convergence for this problem.

(c) Consider starting from the point $\mathbf{x}^{(1)} = [0\ 0]^T$.

i. Verify that the gradient and Hessian at $\mathbf{x}^{(1)}$ are

$$\mathbf{g}^{(1)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \quad G^{(1)} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}.$$

ii. Calculate the Newton search direction by solving $G^{(1)}\mathbf{d}^{(1)} = -\mathbf{g}^{(1)}$. Is $\mathbf{d}^{(1)}$ a descent direction for f at $\mathbf{x}^{(1)}$.

iii. Show that the minimizer of $f(\mathbf{x}^{(1)} + \alpha\mathbf{d}^{(1)})$ is $\alpha^{(1)} = 0$, so that a line search using the Newton direction makes no progress from this starting point. What is the cause of this difficulty?

3. For any positive definite symmetric matrix G there exists a unit lower triangular matrix L and a diagonal matrix D with $D_{ii} > 0$ for $i = 1, \ldots, n$ such that $G = LDL^T$. If we let $D^{1/2} = diag(\sqrt{D_{11}}, \ldots, \sqrt{D_{nn}})$ and $\hat{L} = LD^{1/2}$ then $G = \hat{L}\hat{L}^T$, which is known at the Cholesky factorization of G. Unfortunately you cannot find an LDL^T factorization for every symmetric matrix unless you allow D to have 2×2 blocks on its diagonal (the Bunch-Kaufman factorization). Let

$$\mathbf{g} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \qquad G = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}.$$

(a) Show that G does not have an LDL^T factorization where D is diagonal.

(b) Find all ν such that $G + \nu I$ is positive definite.

(c) [H] Solve $(G + \nu I)\mathbf{d} = -\mathbf{g}$ and express the resulting search direction $\mathbf{d}$ as a linear combination of the steepest descent direction and the Newton direction.

4. The following problem arises in the trust region modification of Newton's method. Let $q(\mathbf{d}) = f_0 + \mathbf{d}^T\mathbf{g}_0 + \frac{1}{2}\mathbf{d}^T G\mathbf{d}$, where f is a constant, $\mathbf{g}_0$ is a constant n-vector and G is a constant $n \times n$ symmetric matrix. Consider the problems

$$(P1) \quad \underset{\mathbf{d} \in \mathbb{R}^n}{Minimize} \quad q(\mathbf{d}) \qquad Subject\ to \quad ||\mathbf{d}||_2 \le \delta$$

and

$$(P2) \quad Solve \quad (G + \nu I)\mathbf{d} = -\mathbf{g}_0.$$

Remember that $||\mathbf{d}||_2 \le \delta \iff \mathbf{d}^T\mathbf{d} \le \delta^2$.

(a) Show that if the constraint in $(P1)$ is NOT active then the solution of $(P1)$ is a solution of $(P2)$ with $\nu = 0$. When is the solution of $(P2)$ with $\nu = 0$ a solution of $(P1)$?

(b) Show that if the constraint in $(P1)$ is active then there exists a $\nu \ge 0$ such that the solution of $(P1)$ is a solution of $(P2)$.

(c) Show that ν is a non-negative root of

$$\phi(\nu) = \mathbf{g}_0^T (G + \nu I)^{-2} \mathbf{g}_0 - \delta^2.$$

(d) Solve $(P1)$ and find the corresponding value of ν when

$$f_0 = 1, \quad \mathbf{g}_0 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \qquad G = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

and $\delta = 3$. Hint: the positive roots of

$$\phi(\lambda) = 144\lambda^4 + 288\lambda^3 + 56\lambda^2 - 72\lambda + 5$$

are $\lambda_1 = 0.0757029156$ and $\lambda_2 = 0.3372822341$.

5. Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = x^2 + cos(x).$$

(a) Show that $f(x)$ is strictly convex on $\mathbb{R}$.

(b) Show that, starting from $\mathbf{x}^{(1)} = 2\ell\pi$ for any integer ℓ, Newton's method with a step size $\alpha^{(k)} = 1$ for all k, does not converge.

(c) Show how Newton's method with a line search to force a reduction in the objective value performs.

(d) Show how Newton's method with a trust region modification and an initial trust region radius of $\delta^{(1)} = 3\pi$ performs.

5.4 Quasi-Newton methods

One disadvantage of Newton's method is the difficulty and expense of evaluating the Hessian matrix. Quasi-Newton methods, which do not require Hessian evaluations, are an extension of the one-dimensional Secant method. In the multi-variable case the Hessian, and hence any approximations to it, are n by n symmetric matrices. It is possible to use either an approximation $B^{(k)}$ to $\nabla^2 f(\mathbf{x}^{(k)})$, or an approximation $H^{(k)}$ to $\nabla^2 f(\mathbf{x}^{(k)})^{-1}$. An important feature of quasi-Newton methods is that the approximations $B^{(k)}$ or $H^{(k)}$ are positive definite. Let $\mathbf{g}^{(k)} \equiv \nabla f(\mathbf{x}^{(k)})$. Then a convex quadratic model

$$q_k(\mathbf{d}) = f^{(k)} + \mathbf{d}^T \mathbf{g}^{(k)} + \tfrac{1}{2}\mathbf{d}^T B^{(k)} \mathbf{d}$$

produces a search direction $\mathbf{d}^{(k)}$ as the solution to the linear system

$$B^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}. \tag{5.4.1}$$

Quasi-Newton methods were originally called variable metric methods [Dav59] as any positive definite matrix B defines a norm (metric) by $\|\mathbf{x}\|_B = (\mathbf{x}^T B \mathbf{x})^{\frac{1}{2}}$. Newton's method in Section 5.3 uses the solution to $\nabla^2 f(\mathbf{x})\mathbf{d} = -\nabla f(\mathbf{x}^{(k)})$, but can have difficulties when the Hessian is not positive definite.

At first it is attractive to use an approximation $H^{(k)}$ to the inverse Hessian, as then the search direction is given by

$$\mathbf{d}^{(k)} = -H^{(k)}\mathbf{g}^{(k)}. \tag{5.4.2}$$

Using (5.4.2) avoids the computational cost of solving the n by n symmetric positive definite linear system (5.4.1) on each iteration. However there are numerical difficulties if $H^{(k)}$ becomes nearly singular (corresponding to $B^{(k)} = {H^{(k)}}^{-1}$ becoming unbounded). These numerical problems are avoided by updating the LDL^T factors of $B^{(k)}$, so the search direction is given by solving

$$L^{(k)} D^{(k)} {L^{(k)}}^T \mathbf{d} = -\mathbf{g}^{(k)}$$

by forward and back substitution. It is important that the LDL^T factorization is *not* recalculated on each iteration, but that the factors $L^{(k)}$ and $D^{(k)}$ are updated instead of $B^{(k)}$. A good discussion of matrix factorizations and the numerical implementation of quasi-Newton methods is in Gill, Murray and Wright [GMW81, GMW91].

To get an understanding of quasi-Newton methods they are discussed in in terms of updating a Hessian approximation $B^{(k)}$.

Algorithm 5.4.1 (Quasi-Newton Method)

1. **Initialization:** *Given a starting point* $\mathbf{x}^{(1)}$*, a symmetric positive definite initial Hessian approximation* $B^{(1)}$*, a tolerances* $\epsilon_{\mathbf{d}} > 0$ *and* $\epsilon_f > 0$ *and a maximum number of iterations* k_{max}*, set* $k = 1$.
2. *Calculate the objective value* $f^{(k)} = f(\mathbf{x}^{(k)})$ *and gradient* $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
3. *Do*
   (a) **Search direction:** *Calculate the quasi-Newton direction by solving* $B^{(k)}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$.
   (b) **Line search:** *Calculate an (exact or approximate) minimizer* $\alpha^{(k)}$ *of* $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$*, starting with* $\alpha^{(k)} = 1$*, to find*
      i. **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$.
      ii. **New function value and gradient:** $f^{(k+1)} = f(\mathbf{x}^{(k+1)})$, $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$.
   (c) **Update Hessian approximation:** $B^{(k+1)} = B^{(k)} + E^{(k)}$.
   (d) *Increase iteration counter:* $k = k + 1$.
4. *Until* $k > k_{max}$ *or* $\|\mathbf{d}^{(k)}\| < \epsilon_{\mathbf{d}}$ *or* $f^{(k)} - f^{(k+1)} < \epsilon_f$.

The major advantage of quasi-Newton methods over Newton methods is that only the function value and gradient must be calculated. If $B^{(k)}$ is positive definite and $\mathbf{g}^{(k)} \neq 0$ then $\mathbf{d}^{(k)}$ is a descent direction, so the line search can produce a $\alpha^{(k)} > 0$ with $f^{(k+1)} < f^{(k)}$.

5.4.1 Updating Hessian approximations

The main question in Algorithm 5.4.1 is how to efficiently update the Hessian approximation $B^{(k)}$. In the Secant method (see Section 4.3) the second derivative is approximated by differences in the first derivative, namely

$$f''(x^{(k)}) \approx \frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Quasi-Newton methods build up curvature information using gradient differences. If $f \in C^2$ a Taylor series expansion of the gradient about $\mathbf{x}^{(k)}$ gives

$$\nabla f(\mathbf{x}^{(k+1)}) \approx \nabla f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}). \tag{5.4.3}$$

Let

$$\mathbf{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \alpha^{(k)}\mathbf{d}^{(k)}, \tag{5.4.4}$$

$$\mathbf{y}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}). \tag{5.4.5}$$

Using (5.4.3), (5.4.4) and (5.4.5) the curvature of the function f at $\mathbf{x}^{(k)}$ in the direction $\mathbf{s}^{(k)}$ can be approximated by

$$\mathbf{s}^{(k)^T}\nabla^2 f(\mathbf{x}^{(k)})\mathbf{s}^{(k)} \approx \mathbf{s}^{(k)^T}\left(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right) = \mathbf{s}^{(k)^T}\mathbf{y}^{(k)}. \tag{5.4.6}$$

If f is a quadratic function then (5.4.6) is exact.

As $B^{(k)}$ must be known before $\mathbf{d}^{(k)}$ and hence $\mathbf{x}^{(k+1)}$ and $\mathbf{g}^{(k+1)}$ can be calculated it is not possible to get $B^{(k)}$ to satisfy (5.4.6). The updated Hessian $B^{(k+1)}$ should approximate the curvature of f along $\mathbf{s}^{(k)}$. This done by requiring $B^{(k+1)}$ to satisfy the *quasi-Newton relation*

$$B^{(k+1)}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}. \tag{5.4.7}$$

This is the fundamental way in which curvature information is included in the Hessian approximations.

As the Hessian is symmetric the approximations $B^{(k)}$ must also be symmetric, so

$$B^{(k+1)^T} = B^{(k+1)}.$$

The quasi-Newton relation (5.4.7) only includes second order information along the direction $\mathbf{d}^{(k)}$, so $B^{(k+1)}$ should only differ from $B^{(k)}$ by a matrix of low rank. A symmetric rank-1 update can be written as

$$B^{(k+1)} = B^{(k)} + \eta \mathbf{u}\mathbf{u}^T,$$

where $\eta \in \mathbb{R}$ and $\mathbf{u} \in \mathbb{R}^n$. There is only one symmetric rank-1 update which satisfies the quasi-Newton relation, namely

$$B^{(k+1)} = B^{(k)} + \frac{1}{\left(\mathbf{y}^{(k)} - B^{(k)}\mathbf{s}^{(k)}\right)^T \mathbf{s}^{(k)}} \left(\mathbf{y}^{(k)} - B^{(k)}\mathbf{s}^{(k)}\right)\left(\mathbf{y}^{(k)} - B^{(k)}\mathbf{s}^{(k)}\right)^T. \tag{5.4.8}$$

Unfortunately the denominator $\left(\mathbf{y}^{(k)} - B^{(k)}\mathbf{s}^{(k)}\right)^T \mathbf{s}^{(k)}$ in (5.4.8) may be zero, so $B^{(k+1)}$ is not defined. Even if the denominator is non-zero the updated Hessian approximation $B^{(k+1)}$ may not be positive definite.

A symmetric rank-2 update can be written as

$$B^{(k+1)} = B^{(k)} + \eta \mathbf{u}\mathbf{u}^T + \zeta \mathbf{v}\mathbf{v}^T, \tag{5.4.9}$$

where $\eta, \zeta \in \mathbb{R}$ and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$. There are a number of rank-2 updates which satisfy the quasi-Newton relation (5.4.7). An important family of such updates is the *Broyden family*

$$B^{(k+1)} = B^{(k)} + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\mathbf{s}^{(k)^T}\mathbf{y}^{(k)}} - \frac{B^{(k)}\mathbf{s}^{(k)}\mathbf{s}^{(k)^T}B^{(k)}}{\mathbf{s}^{(k)^T}B^{(k)}\mathbf{s}^{(k)}} + \phi\left(\mathbf{s}^{(k)^T}B^{(k)}\mathbf{s}^{(k)}\right)\mathbf{w}^{(k)}\mathbf{w}^{(k)^T} \tag{5.4.10}$$

where

$$\mathbf{w}^{(k)} = \frac{\mathbf{y}^{(k)}}{\mathbf{s}^{(k)^T}\mathbf{y}^{(k)}} - \frac{B^{(k)}\mathbf{s}^{(k)}}{\mathbf{s}^{(k)^T}B^{(k)}\mathbf{s}^{(k)}} \tag{5.4.11}$$

and $\phi \in [0,1]$. The DFP update, corresponding to $\phi = 1$, was discovered by Davidon [Dav59] in 1959 and popularised by Fletcher and Powell [FP63]) in 1963. The BFGS, corresponding to $\phi = 0$, was discovered by Broyden [Bro70], Fletcher [Fle70], Goldfarb [Gol70] and Shanno [Sha70a] in 1970. It is now widely believed that the BFGS update is the most effective update from the Broyden family.

5.4.2 Properties

A key property of the Hessian approximation $B^{(k)}$ is positive definiteness. This can be guaranteed if $B^{(1)}$ is positive definite and the quasi-Newton updating formula has the property that if $B^{(k)}$ is positive definite then $B^{(k+1)}$ is positive definite. This can be shown to hold for the Broyden family (5.4.10) if $\phi \in [0,1]$ and $\mathbf{s}^{(k)^T}\mathbf{y}^{(k)} > 0$. The latter condition can be satisfied by performing a sufficiently accurate approximate line search. If $\mathbf{d}^{(k)}$ is a descent direction then $\alpha^{(k)} > 0$ and $-\mathbf{d}^{(k)^T}\mathbf{g}^{(k)} > 0$. An approximate line search controls the accuracy of the line search using the parameter σ where $0 < \sigma < 1$ and the condition

$$|\mathbf{d}^{(k)^T}\mathbf{g}^{(k+1)}| \leq -\sigma \mathbf{d}^{(k)^T}\mathbf{g}^{(k)} \tag{5.4.12}$$

(see Section 5.1.3). As $\sigma < 1$, $\mathbf{d}^{(k)^T}\mathbf{g}^{(k)} < 0$, equation (5.4.12) implies that

$$\mathbf{d}^{(k)^T}\mathbf{g}^{(k)} < \sigma \mathbf{d}^{(k)^T}\mathbf{g}^{(k)} < \mathbf{d}^{(k)^T}\mathbf{g}^{(k+1)}.$$

As $\alpha^{(k)} > 0$ this implies that

$$\mathbf{s}^{(k)^T}\mathbf{y}^{(k)} = \alpha^{(k)}\left(\mathbf{d}^{(k)^T}\mathbf{g}^{(k+1)} - \mathbf{d}^{(k)^T}\mathbf{g}^{(k)}\right) > 0.$$

Another important property of quasi-Newton methods (an other methods) is their ability to find the exact minimizer of a positive definite quadratic function in a finite number of iterations.

Definition 5.4.2 (Quadratic termination) *A line search method has* quadratic termination *if, using exact arithmetic and exact line searches, it finds the minimizer of a strictly convex quadratic function of n variables in at most n iterations.*

Note that a quadratic function is strictly convex if and only if its Hessian (which does not depend on the variables $\mathbf{x}$) is positive definite. A key to quadratic termination is the concept of conjugate directions.

Definition 5.4.3 *The directions $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(n)}$ in $\mathbb{R}^n$ are* conjugate *with respect to a symmetric positive definite matrix G if*

$$\mathbf{d}^{(i)^T} G \mathbf{d}^{(j)} = 0 \quad \textit{for all } i \neq j, i, j = 1, \ldots, n. \tag{5.4.13}$$

Question 9 in Exercises 5.7.6 establishes that conjugate directions are linearly independent.

Theorem 5.4.4 *When applied to a strictly convex quadratic function q of n variables with Hessian G a method which generates search directions which are conjugate with respect to G and uses exact line searches will find the minimizer of q in at most n iterations.*

Moreover $\mathbf{x}^{(k+1)}$ is the minimizer of q in the manifold generated by $\mathbf{x}^{(1)}$ and the search directions $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(k)}$.

Proof The first step is to show that $\mathbf{d}^{(j)^T}\mathbf{g}^{(k)} = 0$ for all $j < k$. As conjugate directions are linearly independent this implies that $\mathbf{g}^{(n+1)} = 0$, so $\mathbf{x}^{(n+1)} = \mathbf{x}^*$. For any $j < k$

$$\begin{aligned} \mathbf{d}^{(j)^T}\mathbf{g}^{(k)} &= \mathbf{d}^{(j)^T}\left(\mathbf{g}^{(k)} - \mathbf{g}^{(k-1)} + \mathbf{g}^{(k-1)} - \mathbf{g}^{(k-2)} + \cdots + \mathbf{g}^{(j+2)} - \mathbf{g}^{(j+1)} + \mathbf{g}^{(j+1)}\right) \\ &= \mathbf{d}^{(j)^T}\left(\mathbf{y}^{(k-1)} + \mathbf{y}^{(k-2)} + \cdots \mathbf{y}^{(j+1)} + \mathbf{g}^{(j+1)}\right). \end{aligned} \tag{5.4.14}$$

For a quadratic function with Hessian G

$$\mathbf{y}^{(i)} = G\mathbf{s}^{(i)} = \alpha^{(i)} G \mathbf{d}^{(i)}.$$

As the directions $\mathbf{d}^{(i)}$ are conjugate with respect to G

$$\mathbf{d}^{(j)^T} G \mathbf{d}^{(i)} = 0 \Longrightarrow \mathbf{d}^{(j)^T}\mathbf{y}^{(i)} = 0 \quad \forall i \neq j.$$

Also an exact line search implies $\mathbf{d}^{(j)^T}\mathbf{g}^{(j+1)} = 0$, so (5.4.14) implies that $\mathbf{d}^{(j)^T}\mathbf{g}^{(k)} = 0$ for all $j < k$. ∎

The quasi-Newton relation $B^{(k+1)}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}$ forces the Hessian approximation to have a particular curvature in the direction $\mathbf{s}^{(k)}$. An important property of the Broyden family of updates is that on a strictly convex quadratic function with Hessian G and using exact line searches then

$$\mathbf{s}^{(i)^T} G \mathbf{s}^{(j)} = 0 \quad \forall\ i \neq j \tag{5.4.15}$$

$$B^{(k)}\mathbf{s}^{(i)} = \mathbf{y}^{(i)} \quad \forall\ i < k. \tag{5.4.16}$$

(5.4.15) is the conjugacy of the search directions $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$ as $\mathbf{s}^{(k)} = \alpha^{(k)}\mathbf{d}^{(k)}$. Equation (5.4.16) states that the later updates do not destroy the curvature information built up on earlier iterations. Let $S = [\ \mathbf{s}^{(1)} \quad \cdots \quad \mathbf{s}^{(n)}\ ]^T$. Then $G\mathbf{s}^{(k)} = \mathbf{y}^{(k)}$ implies that

$$B^{(n+1)}S = GS.$$

If S is non-singular, for example if $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(n)}$ are conjugate, then $B^{(n+1)} = G$.

If f is twice continuously differentiable and an exact line search is made on each iteration then all members of the Broyden family of quasi-Newton updates generate the same sequence of points (Dixon [Dix72], 1972). However the number of function evaluations required by members of the Broyden family can differ significantly. This is attributed to the fact that an initial step $\alpha = 1$ in the line search is closer to the minimum along the line for some updates.

Consider now the case when the objective function f is twice continuously differentiable, the Hessian $\nabla^2 f(\mathbf{x})$ is bounded, and the level set $\{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq f(\mathbf{x}^{(1)})\}$ is bounded. Assuming that the condition

number $\kappa(B^{(k)}) = \|B^{(k)}\|/\|{B^{(k)}}^{-1}\|$ is bounded for all k and an exact line search is used, then quasi-Newton methods can be shown to be globally convergent to a stationary point. The Broyden family of updates does not guarantee that $\kappa(B^{(k)})$ is bounded, however this can be enforced using the LDL^T factorization of $B^{(k)}$.

If $\mathbf{x}^*$ is a local minimizer of f with $\nabla^2 f(\mathbf{x}^*)$ positive definite and $\{\mathbf{x}^{(k)}\} \to \mathbf{x}^*$ then the rate of convergence is superlinear for the DFP and BFGS updates if $\alpha^{(k)} = 1$ for all $k \geq K$. This is one motivation for trying $\alpha = 1$ first in the line search.

Further properties of quasi-Newton methods can be found in Fletcher [Fle87] and Dennis and Moreé [DM77].

Example 5.4.5 (Quasi-Newton method on a quadratic function)
Minimize the function $f(\mathbf{x}) = x_1^2 + 10x_2^2$, starting from $\mathbf{x}^{(1)} = [\,1 \quad \frac{1}{10}\,]^T$, using the BFGS method with $B^{(1)} = I$ and exact line searches. Use exact arithmetic (`Maple` could help) and check $B^{(n+1)} = G$.

5.4.3 Exercises

1. Starting with $\mathbf{x}^{(1)} = [\,0 \quad 0\,]^T$ and $B^{(1)} = I$ use the BFGS method (i.e. a quasi-Newton method using the BFGS update) with exact line searches to minimize

$$f(x) = x_1^2 - 2x_1x_2 + 5x_2^2 + 3x_1 - 12x_2 - 22.$$

   Use exact arithmetic and check that $B^{(3)} = \nabla^2 f$. `Maple` can help with the exact arithmetic.

2. Consider a symmetric rank-1 update $B^{(k+1)} = B^{(k)} + \xi \mathbf{u}\mathbf{u}^T$ where $\xi \in \mathbb{R}$, $\mathbf{u} \in \mathbb{R}^n$ and $B^{(k)}$ is an $n \times n$ symmetric matrix. Show that if $B^{(k+1)}$ satisfies the quasi-Newton relation $B^{(k+1)}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}$ then you obtain the rank-1 updating formula.

3. Show that the quasi-Newton relation $B^{(k+1)}\mathbf{s}^{(k)} = \mathbf{y}^{(k)}$ is satisfied for any member of the Broyden family.

4. [**H**] Prove that if $B^{(k)}$ is positive definite and ${\mathbf{s}^{(k)}}^T\mathbf{y}^{(k)} > 0$ then any quasi-Newton update $B^{(k+1)}$ from the Broyden family with $\phi \in [0,1]$ is positive definite.

   Hint: Show that $\mathbf{z}^T B^{(k+1)}\mathbf{z} > 0$ for any $\mathbf{z} \neq 0$ by showing that

$$\mathbf{z}^T\left(B^{(k)} - \frac{B^{(k)}\mathbf{s}^{(k)}{\mathbf{s}^{(k)}}^T B^{(k)}}{{\mathbf{s}^{(k)}}^T B^{(k)}\mathbf{s}^{(k)}}\right)\mathbf{z} \geq 0 \quad \text{and} \quad \mathbf{z}^T\left(\frac{\mathbf{y}^{(k)}{\mathbf{y}^{(k)}}^T}{{\mathbf{s}^{(k)}}^T\mathbf{y}^{(k)}}\right)\mathbf{z} \geq 0$$

   and that these two inequalities cannot both be satisfied as equalities.

5. Show that the value

$$\phi_1 = \frac{{\mathbf{s}^{(k)}}^T\mathbf{y}^{(k)}}{(\mathbf{y}^{(k)} - B^{(k)}\mathbf{s}^{(k)})^T\mathbf{s}^{(k)}}$$

   in the Broyden family produces the rank-1 update. Assuming that $B^{(k)}$ is positive definite and that ${\mathbf{s}^{(k)}}^T\mathbf{y}^{(k)} > 0$, show that $\phi_1 \notin [0,1]$.

6. [**H**]Let A be an $n \times n$ symmetric nonsingular matrix, and let

$$\hat{A} = A + RSR^T,$$

   where $R \in \mathbb{R}^{n\times m}$, $S \in \mathbb{R}^{m\times m}$ is nonsingular and $m \leq n$.

   (a) Verify that

$$\hat{A}^{-1} = A^{-1} - A^{-1}RT^{-1}R^TA^{-1} \tag{5.4.17}$$

   where $T = S^{-1} + R^TA^{-1}R$. This is the symmetric version of the Sherman-Morrison-Woodbury formula.

   (b) Let $H^{(k)} = (B^{(k)})^{-1}$ and $H^{(k+1)} = (B^{(k+1)})^{-1}$. Use the formula (5.4.17) to obtain an expression for $H^{(k+1)}$ in terms of $H^{(k)}$, $\mathbf{s}^{(k)}$ and $\mathbf{y}^{(k)}$ when $B^{(k+1)}$ is generated by

i. the DFP formula

ii. the BFGS formula.

(c) Show that these formulae are dual in the sense that the DFP formula for updating $H^{(k)}$ is obtained from the BFGS formula for updating $B^{(k)}$ by replacing $B^{(k)}$ by $H^{(k)}$ and swapping $\mathbf{s}^{(k)}$ and $\mathbf{y}^{(k)}$.

7. Show that an efficient way of calculating the BFGS update is

$$B^{(k+1)} = B^{(k)} + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\alpha^{(k)}\mathbf{y}^{(k)^T}\mathbf{d}^{(k)}} + \frac{\mathbf{g}^{(k)}\mathbf{g}^{(k)^T}}{\mathbf{g}^{(k)^T}\mathbf{d}^{(k)}}.$$

Remember that the quasi-Newton search direction $\mathbf{d}^{(k)}$ satisfies $B^{(k)}\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$ and that $\mathbf{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \alpha^{(k)}\mathbf{d}^{(k)}$.

8. Use the quasi-Newton programs available in the class account to minimize the following functions. Try both strong ($\sigma = 0.1$) and weak ($\sigma = 0.9$) line searches.

(a) $f(x) = x_1^2 + 10x_2^2$ starting from $\mathbf{x}^{(1)} = [1\ 0.1]^T$.

(b) $f(x) = x_1^4 + x_1x_2 + (1 + x_2)^2$ starting from $[\ \frac{3}{4}\quad -\frac{11}{8}\ ]^T$, and from $[\ 0\quad 0\ ]^T$.

9. Let $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$ be non-zero directions conjugate with respect to a positive definite matrix G. Show that $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$ are linearly independent.

5.5 Sums of Squares and Nonlinear Equations

5.5.1 Nonlinear equations

A common problem is to find a solution to a system

$$r_i(\mathbf{x}) = 0 \quad \text{for } i = 1, \ldots, m \tag{5.5.1}$$

of m simultaneous nonlinear equations in n variables $\mathbf{x}$. Because such problems frequently arise in data fitting the functions $r_i(\mathbf{x})$ are called *residuals*. Equation (5.5.1) can be written as

$$\mathbf{r}(\mathbf{x}) = 0 \quad \text{where } \mathbf{r}(\mathbf{x}) = \begin{bmatrix} r_i(\mathbf{x}) \\ \vdots \\ r_m(\mathbf{x}) \end{bmatrix},$$

so $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector valued function of n variables. The three cases are

1. If $m < n$, so there are fewer equations than variables, the system $\mathbf{r}(\mathbf{x}) = 0$ is *under-determined*.

2. If $m = n$, so the number of equations equal the number of variables, the system $\mathbf{r}(\mathbf{x}) = 0$ is *well-determined*.

3. If $m > n$, so there are more equations than variables, the system $\mathbf{r}(\mathbf{x}) = 0$ is *over-determined*.

The problem of solving $\mathbf{r}(\mathbf{x}) = 0$ is equivalent to finding a *global* minimizer of the sum of squares objective

$$f(\mathbf{x}) = \sum_{i=1}^{m} r_i(\mathbf{x})^2 = \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}). \tag{5.5.2}$$

As $f(\mathbf{x})$ is a sum of squares it follows immediately that $f(\mathbf{x}) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Thus any point $\mathbf{x}^*$ with $f(\mathbf{x}^*) = 0$ is a global minimizer of $f(\mathbf{x})$. Moreover $f(\mathbf{x}^*) = 0$ if and only if $\mathbf{r}(\mathbf{x}^*) = 0$. On the other hand if $\mathbf{x}^*$ is a *global* minimizer of $f(\mathbf{x})$ with $f(\mathbf{x}^*) > 0$ then there are no solutions to $\mathbf{r}(\mathbf{x}) = 0$. If it is only known that a point $\bar{\mathbf{x}}$ is a *local* minimizer of $f(x)$ and $f(\bar{\mathbf{x}}) > 0$ then there may still be another point $\mathbf{x}^*$ with with $f(\mathbf{x}^*) = 0$.

It is possible to use any vector norm instead of the sum of squares objective. The problem of solving $\mathbf{r}(\mathbf{x}) = 0$ is equivalent to finding a global minimizer of

$$\underset{\mathbf{d} \in \mathbb{R}^n}{Minimize} \quad \|\mathbf{r}(\mathbf{x})\|.$$

In particular the 1-norm is useful when the residual is derived from data which may contain outliers. Approximation problems are discussed further in Chapter 9.

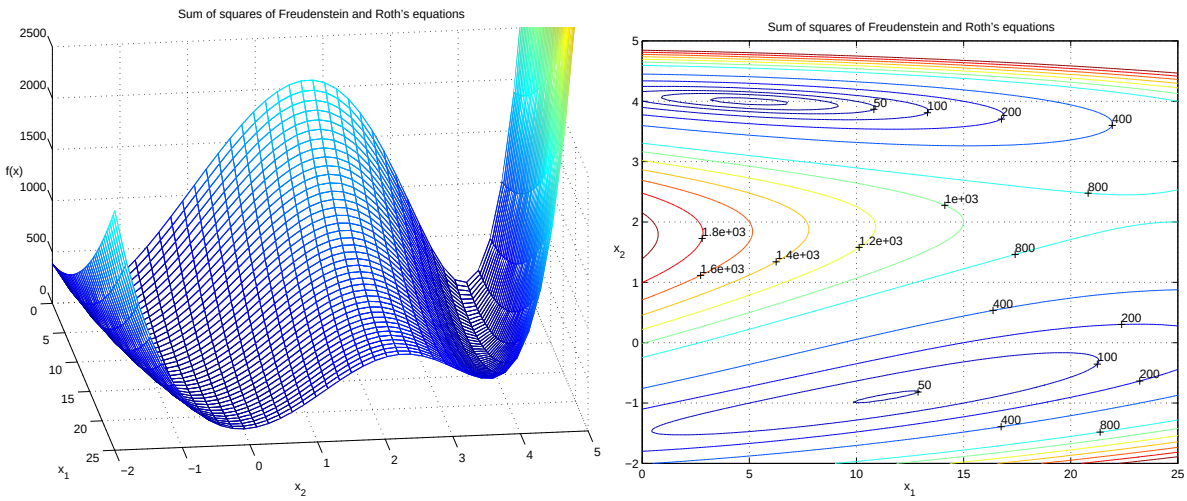


Figure 5.5.1: Surface and contour plots of Freudenstein and Roth's $f(\mathbf{x})$

Example 5.5.1 (Freudenstein and Roth's Equations) *Let $n = 2$, $m = 2$ and*

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} x_1 - x_2^3 + 5x_2^2 - 2x_2 - 13 \\ x_1 + x_2^3 + x_2^2 - 14x_2 - 29 \end{bmatrix}.$$

The sum of squares objective for this well-determined system of nonlinear equations is

$$f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) = 1010 - 84x_1 + 864x_2 + 2x_1^2 + 2x_2^4 + 12x_1x_2^2 - 8x_2^5 + 2x_2^6 - 32x_1x_2 - 80x_2^3 + 12x_2.$$

Surface and contour plots of $f(\mathbf{x})$ are given in Figure 5.5.1. The function f has the global minimizer

$$\mathbf{x}^* = \begin{bmatrix} 5 \\ 4 \end{bmatrix} \text{ with } \mathbf{r}(\mathbf{x}^*) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad f(\mathbf{x}^*) = 0,$$

and a local minimizer

$$\bar{\mathbf{x}} = \frac{1}{3}\begin{bmatrix} 53 - 4\sqrt{22} \\ 2 - \sqrt{22} \end{bmatrix} \approx \begin{bmatrix} 11.412779 \\ -0.8968053 \end{bmatrix}$$

with

$$\mathbf{r}(\bar{\mathbf{x}}) = \frac{4}{27}\begin{bmatrix} 85 - 11\sqrt{22} \\ -85 + \sqrt{22} \end{bmatrix} \approx \begin{bmatrix} 4.948952 \\ -4.948952 \end{bmatrix}, \quad f(\bar{\mathbf{x}}) = \frac{32}{729}(9887 - 1870\sqrt{22}) \approx 48.9842537.$$

Example 5.5.2 (Exponential data fitting) *The rate of decay of radioactive material or a chemical reaction depends on the amount of material or chemical present. If there are two chemicals present the total amount at time t may be given by*

$$\phi(\mathbf{x}; t) = x_1 e^{-x_2 t} + x_3 e^{-x_4 t},$$

where x_1 and x_3 are the starting amounts and x_2 and x_4 are the rate constants. The total amount of chemical is measured at times $t_1, \ldots, t_m$ giving data values $y_i = \phi(\mathbf{x}, t_i)$ for $i = 1, \ldots, m$. One way of estimating the parameters $x_1, \ldots, x_4$ is to form the residuals

$$r_i(\mathbf{x}) = \phi(\mathbf{x}; t_i) - y_i = x_1 e^{-x_2 t_i} + x_3 e^{-x_4 t_i} - y_i$$

for $i = 1, \ldots, m$. Typically there will be more measurements m that variables $\mathbf{x} \in \mathbb{R}^4$, so $\mathbf{r}(\mathbf{x}) = 0$ represents an over-determined system of nonlinear equations. Values for the parameters $\mathbf{x}$ may be estimated by minimizing the sum of squares objective function.

The Jacobian of $\mathbf{r}(\mathbf{x})$ is the matrix $J(\mathbf{x}) \in \mathbb{R}^{m \times n}$ defined by

$$J_{ij}(\mathbf{x}) = \frac{\partial r_i(\mathbf{x})}{\partial x_j}, \quad i = 1, \ldots, m \; j = 1, \ldots, n \qquad \text{so } J(\mathbf{x}) = \begin{bmatrix} \nabla r_1(\mathbf{x})^T \\ \vdots \\ \nabla r_m(\mathbf{x})^T \end{bmatrix}.$$

The gradient of $f(x) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$ is

$$\nabla f(\mathbf{x}) = 2 \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla r_i(\mathbf{x}) = 2 J(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

and the Hessian is

$$\nabla^2 f(\mathbf{x}) = 2 J(\mathbf{x})^T J(\mathbf{x}) + 2 \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla^2 r_i(\mathbf{x}).$$

A classic reference on nonlinear equations is Ortega and Rheinboldt [OR70], while more recent references are Dennis and Schnabel [DS83] and Kelley [Kel95]. Methods for least squares problems are covered by Lawson and Hanson [LH95] and Bjorck [AB96].

5.5.2 Gauss-Newton method

A quasi-Newton (requiring only $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$) or a modified Newton method (requiring $f(\mathbf{x})$, $\nabla f(\mathbf{x})$ and $\nabla^2 f(\mathbf{x})$) could be used to minimize $f(\mathbf{x})$ directly. However the full Hessian is often difficult to obtain. Also a method which just uses $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ does not exploit the full structure of the sum of squares objective. When $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ can be evaluated then the residual vector $\mathbf{r}(\mathbf{x})$ and the Jacobian $J(\mathbf{x})$ can be evaluated. If the residual $r_i(\mathbf{x})$ is small or the Hessians $\nabla^2 r_i(\mathbf{x})$ are small then $H = 2J(\mathbf{x})^T J(\mathbf{x})$ will provide a good approximation to the Hessian of $f(\mathbf{x})$. This approximation could be used in a Newton like method where a step $\mathbf{d}$ is obtained by solving $H\mathbf{d} = -\mathbf{g}$. At $\mathbf{x}^{(k)}$ with $J^{(k)} = J(\mathbf{x}^{(k)})$ and $\mathbf{r}^{(k)} = \mathbf{r}(\mathbf{x}^{(k)})$ this gives

$$J^{(k)^T} J^{(k)} \mathbf{d} = -J^{(k)^T} \mathbf{r}^{(k)}. \tag{5.5.3}$$

The system (5.5.3) is known as the *normal equation* and is very widely used in Statistics. If the problem is over-determined ($m > n$) and $J^{(k)}$ has full rank then $J^{(k)^T} J^{(k)}$ is positive definite (hence nonsingular) and (5.5.3) can be solved for a step $\mathbf{d}^{(k)}$.

Another way of deriving (5.5.3) is to make a first order Taylor series approximation to the residual, giving

$$\mathbf{r}(\mathbf{x}^{(k)} + \mathbf{d}) = \mathbf{r}^{(k)} + J^{(k)} \mathbf{d} + o(\|\mathbf{d}\|).$$

If $\mathbf{d}$ is chosen to try to make $\mathbf{r}(\mathbf{x}^{(k)} + \mathbf{d}) = 0$ then the linear approximation gives

$$J^{(k)} \mathbf{d} = -\mathbf{r}^{(k)}. \tag{5.5.4}$$

However unless the system is well-determined ($m = n$), so $J^{(k)}$ is a square matrix, and $J^{(k)}$ is nonsingular the system (5.5.4) cannot be solved directly to give $\mathbf{d}$. When the system is over-determined both sides of (5.5.4) can be multiplied by $J^{(k)^T}$ to give (5.5.3). In any case $\mathbf{d}^{(k)}$ can be chosen as a solution of the linear least squares problem

$$\underset{\mathbf{d} \in \mathbb{R}^n}{Minimize} \quad ||J^{(k)} \mathbf{d} + \mathbf{r}^{(k)}||_2^2.$$

Expanding the objective function of the linear least squares problem, and setting the gradient to zero produces the normal equations (5.5.3).

Algorithm 5.5.3 (Basic Gauss-Newton method)

1. **Initialization:** *Given a starting point* $\mathbf{x}^{(1)}$*, a tolerance* $\epsilon_g > 0$*, and a maximum number of iterations* k_{max}*, set* $k = 1$*.*
2. *Calculate the residual* $\mathbf{r}^{(k)} = \mathbf{r}(\mathbf{x}^{(k)})$ *and the Jacobian* $J^{(k)} = J(\mathbf{x}^{(k)})$*.*
3. *While* $k < k_{max}$ *and* $\|J^{(k)^T}\mathbf{r}^{(k)}\| > \epsilon_g$
   (a) **Solve the linear least squares problem**
   $$\begin{array}{ll} Minimize & \|J^{(k)}\mathbf{d} + \mathbf{r}^{(k)}\|_2^2 \\ \mathbf{d} \in \mathbb{R}^n & \end{array}$$
   for the step $\mathbf{d}^{(k)}$*.*
   (b) **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$*.*
   (c) **New residual and Jacobian:** $\mathbf{r}^{(k+1)} = \mathbf{r}(\mathbf{x}^{(k+1)})$ *and the Jacobian* $J^{(k+1)} = J(\mathbf{x}^{(k+1)})$*.*
   (d) *Increase iteration counter:* $k = k + 1$*.*

   End while loop.

The main attraction of the Gauss-Newton method is its fast rate of convergence when the minimizer is "nice". A proof of Proposition 5.5.4 can be found in [Fle87, DS83].

Proposition 5.5.4 *If* $\mathbf{x}^*$ *is a local minimizer of* $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T\mathbf{r}(\mathbf{x})$ *where*

1. $J(\mathbf{x}^*)$ *has full rank,*
2. $\sum_{i=1}^m r_i(\mathbf{x}^*)\nabla^2 r_i(\mathbf{x}^*) = 0$

and $\mathbf{x}^{(1)}$ *is sufficiently close to* $\mathbf{x}^*$*, then the iterates* $\{\mathbf{x}^{(k)}\}$ *produced by the basic Gauss-Newton method will converge to* $\mathbf{x}^*$ *and the rate of convergence is quadratic.*

If either of the conditions 1. and 2. of Proposition 5.5.4 are not satisfied then the rate of convergence is at best linear.

Proposition 5.5.4 is only a local result. The standard techniques to try to get a method to converge from any starting point (*global convergence*) are to either add a line search or include a trust region. The basic Gauss-Newton method can easily be modified so the direction $\mathbf{d}^{(k)}$ produced by solving the linear least squares problem is used as a search direction and a step $\alpha^{(k)}$ is calculated by an exact or approximate line search on $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$. The new iterate is then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$. If $m \geq n$ and $J^{(k)}$ has full rank then $J^{(k)^T}J^{(k)}$ is positive definite, so the direction $\mathbf{d}^{(k)}$ produced by the normal equations is a descent direction. However a line search does not address the fact that the direction $\mathbf{d}^{(k)}$ is not well defined if $J^{(k)}$ does not have full rank.

The *Levenberg-Marquadt method* adds a multiple of the identity matrix to the Hessian approximation. The direction $\mathbf{d}^{(k)}$ is then obtained by solving

$$\left(J^{(k)^T}J^{(k)} + \nu^{(k)}I\right)\mathbf{d} = -J^{(k)^T}\mathbf{r}^{(k)},$$

where $\nu^{(k)} \geq 0$. If $J^{(k)}$ has full rank then $\nu^{(k)} = 0$ will produce a descent direction. If $J^{(k)}$ does not have full rank then $\nu^{(k)} > 0$ ensures that $\mathbf{d}^{(k)}$ is well-defined and a descent direction.

Either rules for directly choosing $\nu^{(k)}$ can be implemented and the resulting direction $\mathbf{d}^{(k)}$ used in a line search, or $\nu^{(k)}$ can be defined implicitly by calculating $\mathbf{d}^{(k)}$ as a solution of the trust region problem

$$\begin{array}{ll} Minimize & \|J^{(k)}\mathbf{d} + \mathbf{r}^{(k)}\|_2^2 \\ \mathbf{d} \in \mathbb{R}^n & \\ Subject\ to & \|\mathbf{d}\| \leq \delta^{(k)}. \end{array}$$

It is also important to realize that forming ${J^{(k)}}^T J^{(k)}$ in the normal equations is a numerically unstable process which squares the condition number of the linear system that must be solved. It is preferable to solve the linear least squares problem using a QR factorization of $J^{(k)}$. See Sections A.7.3 and A.8.4 in Appendix A for more details on the QR factorization and solving over-determined linear systems.

5.5.3 Well-determined systems

For a well-determined system, with the same number of variables as equations ($m = n$), Newton's method linearizes the equations $\mathbf{r}(\mathbf{x}) = 0$, giving

$$\mathbf{r}(\mathbf{x}^{(k)} + \mathbf{d}) = \mathbf{r}^{(k)} + J^{(k)}\mathbf{d} + o(\|\mathbf{d}\|),$$

where $J^{(k)} = J(\mathbf{x}^{(k)})$ and $\mathbf{r}^{(k)} = \mathbf{r}(\mathbf{x}^{(k)})$. The step $\mathbf{d}$ is chosen to try to make $\mathbf{r}(\mathbf{x} + \mathbf{d}) = 0$, giving the Newton direction $\mathbf{d}_N^{(k)}$ as the solution to

$$J^{(k)}\mathbf{d} = -\mathbf{r}^{(k)}. \tag{5.5.5}$$

In the well-determined case the Jacobian $J^{(k)}$ is a square n by n matrix, but is in general not symmetric. If the Jacobian is nonsingular then the Newton direction $\mathbf{d}_N^{(k)} = -\left(J^{(k)}\right)^{-1}\mathbf{r}^{(k)}$ exists. In practice an LU factorization is used to solve (5.5.5).

If $\mathbf{x}^*$ is a solution to the system of equations, so $\mathbf{r}(\mathbf{x}^*) = 0$, the Jacobian J^* is nonsingular, and if $\mathbf{x}^{(1)}$ is *sufficiently close* to $\mathbf{x}^*$, then the Newton direction is will defined ($J^{(k)}$ is nonsingular for all k) and the basic Newton method $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}_N^{(k)}$ converges to $\mathbf{x}^*$ and the rate of convergence is quadratic. If the Jacobian J^* is singular then Newton's method may still converge to $\mathbf{x}^*$ but with a linear rate of convergence or Newton's method may fail to converge to $\mathbf{x}^*$.

Just what the statement "$\mathbf{x}^{(1)}$ is sufficiently close to $\mathbf{x}^*$" means in practice is not clear, except that it is important to use a good a starting point $\mathbf{x}^{(1)}$. The set of starting points which converge to a particular solution $\mathbf{x}^*$ is the *domain of attraction* for $\mathbf{x}^*$. The domain of attraction can be a wide variety of shapes, including *fractal sets* in which the boundaries have complex structure.

Example 5.5.5 (Newton's method and fractals) *Let $z \in \mathbb{C}$ and consider the system of equations*

$$r(\mathbf{z}) = z^3 - 1 = 0 \tag{5.5.6}$$

where $r : \mathbb{C} \to \mathbb{C}$. This is equivalent (see Exercise 8) to the system

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} x_1^3 - 3x_1x_2^2 - 1 \\ 3x_1^2x_2 - x_2^3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.5.7}$$

of two equations in two variables. The solutions to this are

$$\mathbf{x}^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{x}} = \begin{bmatrix} -\frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}, \quad \overline{\mathbf{x}} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix}.$$

At each of these points the Jacobian is nonsingular, so Newton's method will exhibit a quadratic rate of convergence if an iterate falls in the domain of attraction of a root. The set of points which do not converge to $\mathbf{x}^*$ in 30 iterations is illustrated in Figure 5.5.2. Equally interesting is to look at the number of iterations that Newton's method takes to converge to one of the roots $\mathbf{x}^*, \hat{\mathbf{x}}, \overline{\mathbf{x}}$. Figure 5.5.3 gives a contour plot of the number of iterations that Newton's method takes to converge from different starting points. Both the plots in Figure 5.5.2 and 5.5.3 are typical of fractal sets.

5.5.4 Exercises

1. Rosenbrock's equations are

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} 10(x_2 - x_1^2) \\ 1 - x_1 \end{bmatrix}.$$

   (a) Is this a under-determined, well-determined or over-determined problem?

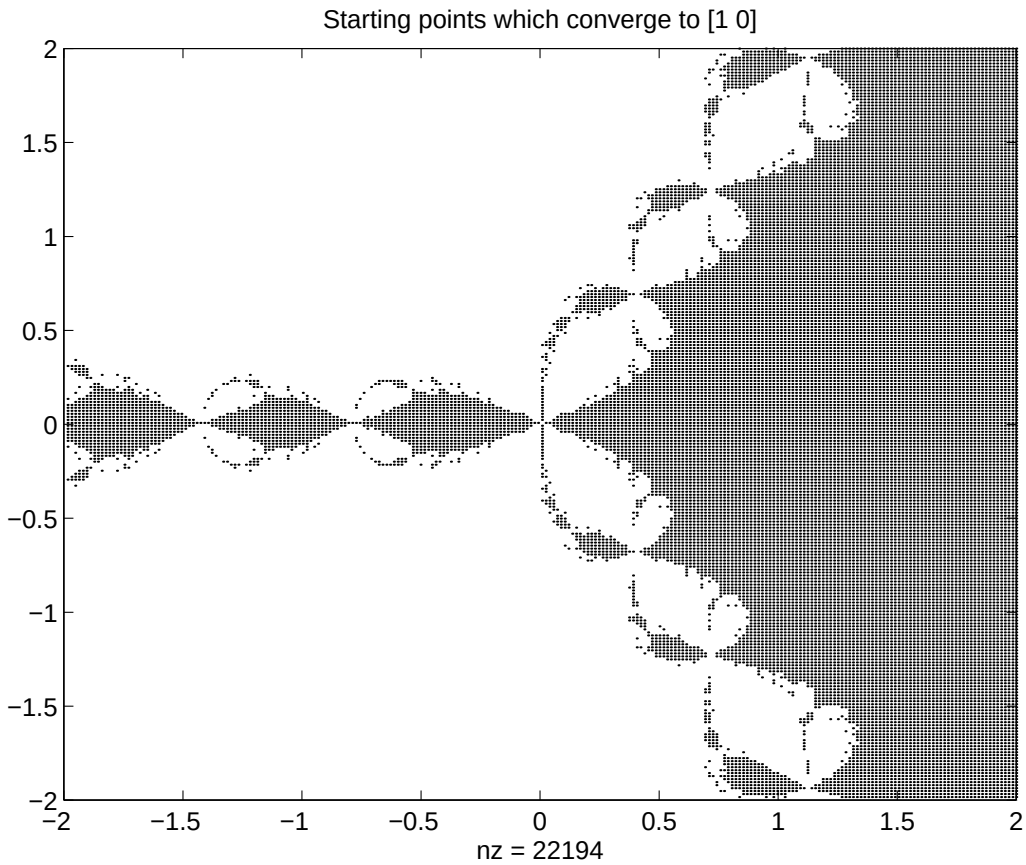


Figure 5.5.2: Domain of attraction of Newton's method to $\mathbf{x}^*$

(b) Find the sum of squares function $f(\mathbf{x}) = \mathbf{r}^T(\mathbf{x})\mathbf{r}(\mathbf{x})$.

(c) Calculate the Jacobian $J(\mathbf{x})$ of $\mathbf{r}(\mathbf{x})$ and the gradient $\mathbf{g}(\mathbf{x}) \equiv \nabla f(\mathbf{x})$ of $f(\mathbf{x})$. Verify that $\mathbf{g}(\mathbf{x}) = 2J(\mathbf{x})^T\mathbf{r}(\mathbf{x})$.

(d) Show that the Jacobian $J(\mathbf{x})$ has full rank for all $\mathbf{x} \in \mathbb{R}^n$.

(e) Calculate the Gauss-Newton approximation to the Hessian, and show it is positive definite for all $\mathbf{x} \in \mathbb{R}^n$.

(f) Calculate the Hessian $G(\mathbf{x}) \equiv \nabla^2 f(\mathbf{x})$ of $f(\mathbf{x})$ and verify that

$$G(\mathbf{x}) = 2J^T(\mathbf{x})J(\mathbf{x}) + 2\sum_{i=1}^{m} r_i(\mathbf{x})\nabla^2 r_i(\mathbf{x}).$$

(g) When will the Gauss-Newton approximation to the Hessian be exact?

(h) Are the following points strict local or global minimizers of $f(\mathbf{x})$?

  i. $\bar{\mathbf{x}} = [\ 0 \quad 1\ ]^T$.

  ii. $\mathbf{x}^* = [\ 1 \quad 1\ ]^T$.

(i) Rosenbrock's function is a standard test problem for numerical methods for unconstrained optimization. The starting point is $\mathbf{x}^{(1)} = [\ -1.2 \quad 1\ ]^T$.

  i. Compare the steepest descent method with the Fletcher-Reeves and Polak-Ribiere conjugate gradient methods using a strong approximate line search ($\rho = 0.01$, $\sigma = 0.1$).

  ii. Compare the DFP and BFGS quasi-Newton methods, using a weak approximate line search ($\rho = 0.01$, $\sigma = 0.9$). What would you expect if exact line searches are used?

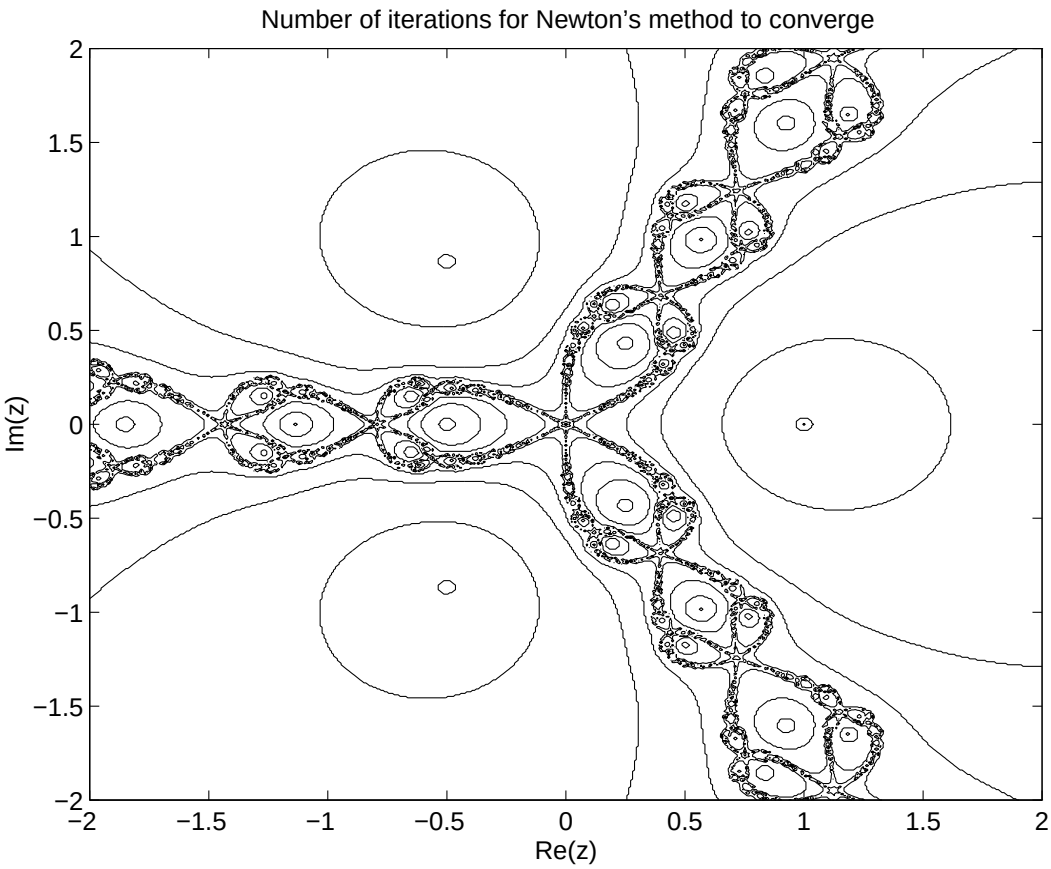


Figure 5.5.3: Number of iterations for Newton's method to converge

iii. Compare the modified Newton method, trust region (restricted step) Newton method and the Gauss-Newton method using (where appropriate) a weak approximate line search ($\rho = 0.01$, $\sigma = 0.9$).

iv. Summarize the relative strengths and weaknesses of the above methods (steepest descent, conjugate gradient, quasi-Newton, modified Newton, trust region Newton, and Gauss-Newton).

2. Let $\mathbf{r} : \mathbb{R} \to \mathbb{R}^2$ be defined by

$$\mathbf{r}(x) = \begin{bmatrix} x+1 \\ \xi x^2 + x - 1 \end{bmatrix},$$

where $\xi \in \mathbb{R}$ is a fixed parameter.

(a) Show that if $\xi < 1$ then $x^* = 0$ is a strict local minimizer of the sum of squares objective function. Does x^* solve $\mathbf{r}(x) = 0$?

(b) Show that

i. the correction term $d^{(k)}$ in the basic Gauss-Newton method is

$$d^{(k)} = -\frac{x^{(k)}\left(2\xi^2 {x^{(k)}}^2 + 3\xi x^{(k)} + 2 - 2\xi\right)}{2\left(2\xi^2 {x^{(k)}}^2 + 2\xi x^{(k)} + 1\right)}.$$

ii. the iterates generated by the basic Gauss-Newton method satsify

$$\frac{x^{(k+1)}}{x^{(k)}} = \xi \frac{2\xi {x^{(k)}}^2 + x^{(k)} + 2}{2(2\xi^2 {x^{(k)}}^2 + 2\xi x^{(k)} + 1)}.$$

iii. if $x^{(k)} \to x^* = 0$ and $|\xi| < 1$ then the rate of covergence is linear with rate constant $\beta = |\xi|$.

(c) With $\xi = 0.1$ apply the Gauss-Newton method starting at $x^{(1)} = 1$ (see the MATLAB file `ssq2.m`).

i. Stop when $\|g^{(k)}\| < 1 \times 10^{-15}$, and estimate the rate of convergence.

ii. When would the iteration have stopped with the convergence test $\|d^{(k)}\| < 1 \times 10^{-10}$?

(d) With $\xi = -2$ apply the basic Gauss-Newton method starting at $x^{(1)} = 1$. What is happening?

(e) Use Newton's method, with step size $\alpha^{(k)} = 1\ \forall k$, to minimize $f(x)$ starting from $x^{(1)} = 1$ for $\xi = 0.1$ and $\xi = -2$, and compare your results the Gauss-Newton method (see the MATLAB file `ssq2.m`).

3. (a) Let J be an $m \times n$ matrix where $m \geq n$.

i. Show that if J has full rank then $J^T J$ is positive definite.

ii. Give an example of a non-zero matrix J for which $J^T J$ is not positive definite.

(b) Let

$$J = \begin{bmatrix} 1.000 & 1.020 \\ 1.000 & 1.000 \\ 1.000 & 1.000 \end{bmatrix}$$

Working to four decimal digits, calculate $J^T J$ and $\det J^T J$. Show that even though J has full rank, the calculated matrix $J^T J$ is singular when working with 4 significant digits.

4. Calculate the least squares solution to the system of linear equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ where

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} 3x_1 - 4x_2 - 2 \\ 3x_1 - 4x_2 + 3 \\ 6x_2 - 1 \end{bmatrix}.$$

(a) using the generalized inverse $J^+ = (J^T J)^{-1} J^T$.

(b) using an orthogonal factorization so $J^+ = R^{-1} Q_1^T$ where

$$Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & -1 & 0 \end{bmatrix}.$$

5. Show that $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$ where

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} x_1 - 1 \\ x_1 + 1 \end{bmatrix},$$

has a unique global minimizer $\mathbf{x}^*$ but that $\mathbf{r}(\mathbf{x}^*) \neq \mathbf{0}$.

6. The following model arose in a study of the productivity impact of staffing levels when developing computer software. The quantities measured were P = project productivity, L = project size, and S = maximum staff level. The model assumed a relation of the form

$$P = x_1 L^{x_2} S^{x_3},$$

where x_1,x_2 and x_3 are parameters to be determined. Suppose you have data P_i, L_i, S_i for $i = 1, \ldots, m$ from m different projects.

(a) Define the residual $r_i(\mathbf{x})$ for $i = 1, \ldots, m$ and the corresponding sum of squares objective function $f(\mathbf{x})$. Calculate the Jacobian matrix $J(\mathbf{x})$ of $\mathbf{r}(\mathbf{x})$ and the gradient $\mathbf{g}(\mathbf{x}) \equiv \nabla f(\mathbf{x})$ of $f(\mathbf{x})$.

(b) If you can evaluate first derivatives but not second derivatives for this problem which are the most suitable methods for minimizing f. Briefly summarize the advantages and disadvantages of these methods.

(c) Use the MATLAB least squares routine `lsqnonlin` to solve this problem for the data

i	1	2	3	4	5
P_i	450	830	547	628	1508
L_i	3539	42487	12620	21698	30521
S_1	3	7	5	6	4

(d) Find a transformation of the problem that produces a very much simpler optimization problem. Solve the transformed problem using an appropriate MATLAB routine.

(e) Comment on the solution you obtain, in particular the significance of the value of x_3^*, the exponent of the maximum staffing level.

7. A model of the Australian sheep population (p) as a function of time (t) is

$$p(t) = x_1 + x_2 t + x_3 \sin(x_4 + x_5 t).$$

You have data[1] p_i at times t_i for $i = 1, \cdots, m$ and you want to find the 'best' estimate the parameters $x_1, \cdots, x_5$.

(a) Discuss how this can be posed as a mathematical optimization problem, and what are the possible definitions of 'best'.

(b) Use MATLAB to find the linear least squares approximation of the form $a_1 + a_2 t$ and the nonlinear function $p(t)$ to fit the data in Table 5.5.1 For the nonlinear problem use the MATLAB

Year	1986	1987	1988	1989	1990	1991
Population	150.390	153.177	156.606	164.811	173.632	166.550
Year	1992	1993	1994	1995	1996	1997
Population	150.582	140.541	132.570	123.210	121.116	123.332

Table 5.5.1: Australian sheep population in millions

`lsqnonlin` routine, starting from $x^{(0)} = \begin{bmatrix} a_1 & a_1 & 1 & 0 & 1 \end{bmatrix}$, where $\mathbf{a}$ is the linear least squares approximation. Plot the data and approximations. Comment on the answers you obtain.

i. Provide only a function to calculate the residual values. In this case MATLAB uses finite differences to approximate the gradients.

ii. Provide both a function for the residuals and a function for the gradients of the residuals. See `help lsqnonlin` for more information. Comment on any differences when the gradients are provided.

(c) The nonlinear approximation $p(t)$ obtained by a least squares fit in the previous part may have oscillations of large amplitudes. See if any of the following strategies, or any others you can think of, produce a better approximation.

i. Other starting points, such as $x^{(0)} = \begin{bmatrix} a_1 & a_1 & 20 & 0 & 0.1 \end{bmatrix}$,

ii. Shifting or scaling the time variable, i.e. transformations of the form $\tau = \alpha t + \beta$.

iii. Using a criteria other that least squares.

iv. Adding a smoothing term to include a norm of the second derivative of $p(t)$ in the objective.

8. [**H**] Let $z \in \mathbb{C}$ and let

$$r(z) = z^3 - 1. \tag{5.5.8}$$

(a) By taking $z = x_1 + x_2\sqrt{-1}$ show that the complex equation $r(z) = 0$ is equivalent to the system

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} x_1^3 - 3x_1x_2^2 - 1 \\ 3x_1^2x_2 - x_2^3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5.5.9}$$

of two equations in two variables.

[1] Australian Commodity Statistics, Australian Bureau of Agricultural and Resource Economics, Canberra

(b) Calculate the Jacobian $J(\mathbf{x})$ of $\mathbf{r}(\mathbf{x})$, and find all points where $J(\mathbf{x})$ is singular.

(c) Show that

$$\mathbf{x}^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{\mathbf{x}} = \begin{bmatrix} -\frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}, \quad \overline{\mathbf{x}} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix},$$

are solutions to $\mathbf{r}(\mathbf{x}) = 0$, and that the Jacobian in nonsingular at each root.

(d) Show that Newton's method in complex variables is

$$z^{(k+1)} = z^{(k)} - \frac{{z^{(k)}}^3 - 1}{3z^{(k)}2}, \tag{5.5.10}$$

which is equivalent to $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}_N^{(k)}$ where

$$\mathbf{d}_N^{(k)} = -\frac{1}{3(x_1^4 + 2x_1^2x_2^2 + x_2^4)} \begin{bmatrix} x_1^5 + 2x_1^3x_2^2 - x_1^2 + x_1x_2^4 + x_2^2 \\ x_1^4x_2 + 2x_1^2x_2^3 + 2x_1x_2 + x_2^5 \end{bmatrix}.$$

If you are using `Maple` it may help to let `z := u + I*v;`, where the commands `assume(u, real);` and `assume(v, real);` have been used to tell `Maple` that `u` and `v` are real.

(e) Write a MATLAB program to perform up to 30 iterations of Newton's method to find a solution of $\mathbf{r}(\mathbf{x}) = 0$ with starting points coming from the region $\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{a} \le \mathbf{x} \le \mathbf{b}\}$. Consider taking N_1 points in the x_1 variable and N_2 points in the x_2 variable, so $\mathbf{x}^{(1)} = [\ u_i \quad v_j\ ]^T$ where

$$\begin{aligned} u_i &= a_1 + ih_1 \quad i = 0, \ldots, N_1, \qquad h_1 = \frac{b_1 - a_1}{N_1}, \\ v_j &= a_2 + ih_2 \quad j = 0, \ldots, N_2, \qquad h_2 = \frac{b_2 - a_2}{N_2}. \end{aligned}$$

Reasonable values to try are

$$\mathbf{a} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad N_1 = 50, \quad N_2 = 50.$$

Generate plots of the starting points which converge to $\mathbf{x}^*$, and the number of iterations Newton's method takes to converge to any one of the roots $\mathbf{x}^*, \overline{\mathbf{x}}, \hat{\mathbf{x}}$.

Hint: It is efficient to iterate on a $N_1 + 1$ by $N_2 + 1$ matrix Z which has complex elements corresponding to the iterates generated from each starting point.

9. [**H**] An engineer has measured the amplitude of a damped oscillator at one second intervals for 39 seconds. The data y_i measured at time t_i, (available in the file `doscd.dat` in the class account), is listed in Table 5.5.2 and plotted in Figure 5.5.4. They wish to fit a function of the form

$$\phi(\mathbf{x}, t) = x_1 + x_2 e^{-x_3 t} \cos(x_4 t + x_5)$$

(a) The company's engineer has defined the residual

$$r_i(\mathbf{x}) = \phi(\mathbf{x}, t_i) - y_i.$$

The optimization programs the company has require the Jacobian $J(\mathbf{x})$ of the residual $\mathbf{r}(\mathbf{x})$, and the gradient $\mathbf{g}(\mathbf{x})$ and Hessian $G(\mathbf{x})$ of the sums of squares objective

$$f(\mathbf{x}) = \sum_{i=1}^{m} \left[r_i(\mathbf{x})^2\right] = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}).$$

Give expressions for these quantities.

Time t_i	Amplitude y_i	Time t_i	Amplitude y_i	Time t_i	Amplitude y_i	Time t_i	Amplitude y_i
0.0	7.75	10.0	6.07	20.0	5.34	30.0	5.44
1.0	6.40	11.0	5.29	21.0	5.16	31.0	4.82
2.0	3.76	12.0	4.74	22.0	4.94	32.0	4.90
3.0	3.83	13.0	4.60	23.0	5.41	33.0	5.35
4.0	5.67	14.0	5.72	24.0	5.27	34.0	5.41
5.0	7.04	15.0	5.62	25.0	5.61	35.0	5.18
6.0	5.45	16.0	5.13	26.0	4.93	36.0	5.18
7.0	4.05	17.0	4.74	27.0	5.24	37.0	5.30
8.0	4.37	18.0	5.24	28.0	5.32	38.0	5.52
9.0	5.80	19.0	5.22	29.0	5.31	39.0	5.36

Table 5.5.2: Oscillator data

(b) Write MATLAB functions

- `[r, J] = doscrj(x, t, y)`
  to calculate the residual `r` and Jacobian `J` at an arbitrary point `x`.
- `[f, g, G] = doscfg(x, t, y)`
  to calculate the sum of squares `f`, its gradient `g` and Hessian `G` at an arbitrary point `x`.

How can they check these routines are correct?

(c) What values of the parameters will produce a function that best fits this data. Use both

i. The MATLAB least squares routine `leastsq`.

ii. A MATLAB unconstrained minimization routine. See `help optim` for a list of available routines.

(d) The engineer tried to solve this problem using a least squares method starting from $\mathbf{x} = 0$, but had lots of numerical problems. Any suggestions?

(e) Another engineer believes that that

$$x_1 \geq \bar{y} \equiv \left(\sum_{i=1}^{m} y_i \right) / m.$$

Would this constraint change the solution?

5.6 Methods using only function values

In many complex applications only the function value $f(\mathbf{x}^{(k)})$ can be evaluated. It is still important to try to determine if the function is at least once continuously differentiable. The basic options when only the function value can be evaluated are

1. If $f(\mathbf{x})$ is know to be twice continuously differentiable then the gradient can be approximated. The approximate gradients can, with care, be used in a method which would normally require gradients, for example a quasi-Newton method.

   (a) Use finite differences to approximate the gradient, for example

   $$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + \eta \mathbf{e}_i) - f(\mathbf{x})}{\eta}.$$

   The choice of $\eta > 0$ depends on balancing the accuracy of the approximation with minimizing the effects of rounding error (see Section 1.4.4 in Chapter 1).

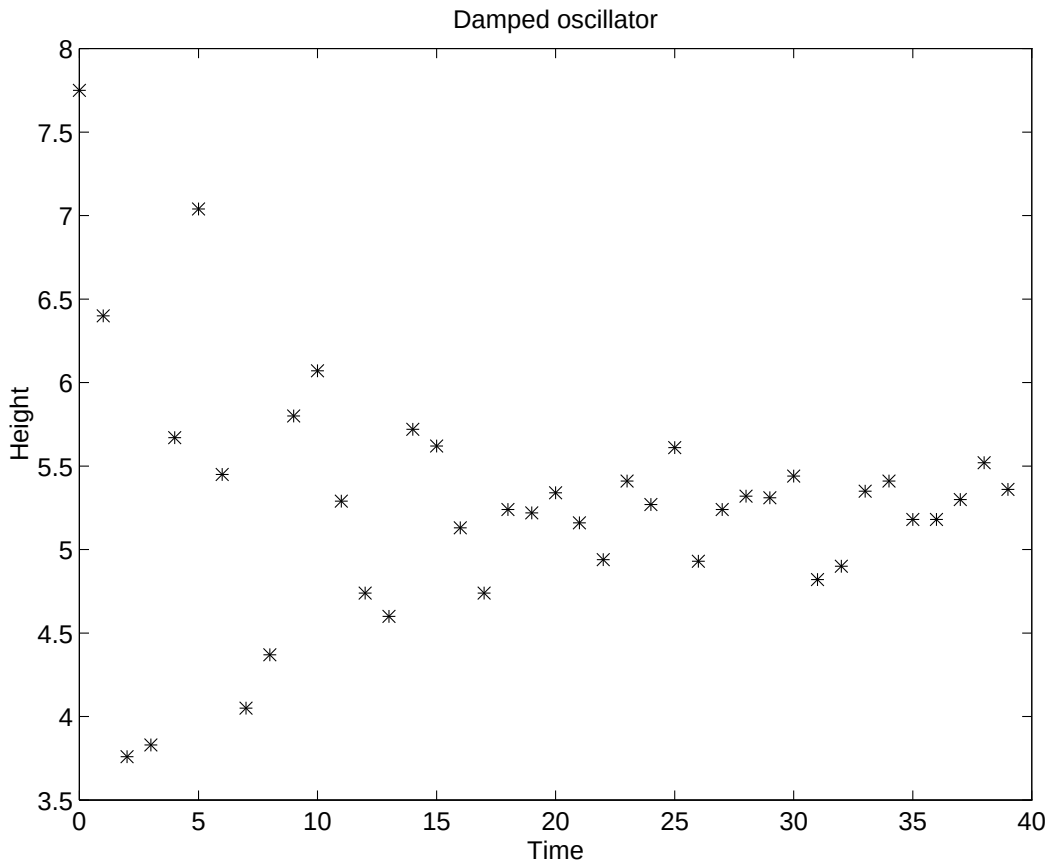


Figure 5.5.4: Oscillator amplitudes at 1 second intervals

(b) If a routine to calculate $f(\mathbf{x})$ is provided in Fortran or C then automatic differentiation [GC91] can be used to generate a routine to calculate the gradient.

2. Use a conjugate direction algorithm, which assumes the function can be modelled by a quadratic function.

3. If the function is only known to be continuous use the Nelder–Meade simplex algorithm which reflects a simplex in $\mathbb{R}^n$ using function values at $n+1$ points.

5.6.1 Conjugate direction methods

Powell's algorithm [Pow64] uses line searches to generate conjugate direction based on the following Theorem.

Theorem 5.6.1 (Parallel subspace property) *Consider two parallel subspaces*

$$S_1 = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{u} + \sum_{i=1}^{k} \eta_i \mathbf{d}^{(i)}, \eta_i \in \mathbb{R} \right\} \tag{5.6.1}$$

$$S_2 = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{v} + \sum_{i=1}^{k} \zeta_i \mathbf{d}^{(i)}, \zeta_i \in \mathbb{R} \right\} \tag{5.6.2}$$

generated by linearly independent directions $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(k)}$*, where* $k < n$ *and* $\mathbf{u} \neq \mathbf{v}$*. Let* $\mathbf{z}_1$ *be the minimizer of a strictly convex quadratic function over* S_1 *and let* $\mathbf{z}_2$ *be the minimizer over* S_2*. Then* $\mathbf{z}_2 - \mathbf{z}_1$ *is conjugate to* $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \ldots, \mathbf{d}^{(k)}$*.*

Proof Let G be the positive definite Hessian of the quadratic function. As $\mathbf{z}_1$ is the minimizer over S_1

$$\nabla f(\mathbf{z}_1)^T \mathbf{d}^{(i)} = 0, i = 1, \ldots, k.$$

As $\mathbf{z}_2$ is the minimizer over S_2

$$\nabla f(\mathbf{z}_2)^T \mathbf{d}^{(i)} = 0, i = 1, \ldots, k.$$

Subtracting gives, for $i = 1, \ldots, k$

$$\left(\nabla f(\mathbf{z}_2) - \nabla f(\mathbf{z}_1)\right)^T \mathbf{d}^{(i)} = 0.$$

As a quadratic function has a linear gradient which satisfies $\nabla f(\mathbf{z}_2) - \nabla f(z_1) = G(\mathbf{z}_2 - \mathbf{z}_1)$ this implies

$$(\mathbf{z}_2 - \mathbf{z}_1)^T G \mathbf{d}^{(i)} = 0 \quad i = 1, \ldots, k.$$

∎

5.6.2 Nelder-Meade Simplex method

5.7 Large scale methods

For problems with a large number of variables n (more than several hundred) the storage of the Hessian matrix $\nabla^2 f(\mathbf{x})$ or a quasi-Newton approximation $B^{(k)}$, requires $\frac{n^2}{2} + O(n)$ elements. Calculating the Newton direction by solving the symmetric positive definite linear system $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$ using a Cholesky factorization requires $\frac{n^3}{2} + O(n^2)$ operations. If the LDL^T factors of $B^{(k)}$ are updated in a quasi-Newton method then solving the linear system $L^{(k)}D^{(k)}{L^{(k)}}^T\mathbf{d} = -\mathbf{g}^{(k)}$ is an $O(n^2)$ operation, as it updating the factors $D^{(k)}$ and $L^{(k)}$ by a quasi-Newton formula. Thus both the storage requirements and the number of calculations per iteration can become limitations when n is very large.

Typically the Hessian of large problems is very sparse. The Hessian may also have additional structure that can be utilized, for example the block diagonal structure arising in a partially separable function (see Example 1.3.1 on page 34).

The main approaches to large scale unconstrained optimization problems include methods which exploit the sparse structure of the Hessian (for example re-orderings of the variables to make the Cholesky factorization more efficient or the use of iterative methods to approximately solve the linear system defining the search direction); enforcing sparsity in the quasi-Newton approximations, using a limited memory quasi-Newton method, or using a conjugate gradient method. These approaches to determining a search direction $\mathbf{d}^{(k)}$ are briefly summarised below. A general reference in this area is Coleman [Col84] which also considers linear equations, linear least squares and linear programming problems, as well as unconstrained problems.

5.7.1 Conjugate gradient methods

A disadvantage of both Newton and quasi-Newton methods, especially when the number of variables n is large, is the need to store the n by n Hessian or an approximation to it. Even making use of the symmetry this requires $n(n+1)/2$ elements to be stored. Conjugate gradient methods were developed initially by Hestenes and Stiefel [HS52] in 1952 as an iterative method for solving positive definite linear systems. Minimizing a positive definite quadratic function $q(\mathbf{x}) = q_0 + \mathbf{g}_0^T x + \frac{1}{2}\mathbf{x}^T G\mathbf{x}$ is equivalent to solving the linear system $\nabla q(\mathbf{x}) = G\mathbf{x} + \mathbf{g}_0 = 0$. Conjugate gradient methods are widely used as an iterative techniques for solving large sparse linear systems when the coefficient matrix is symmetric and positive definite. Conjugate gradient methods require the gradient to be calculated (or equivalently the matrix-vector product $G\mathbf{x}$ to be calculated), but are economical in terms of storage and housekeeping as they require only a few n dimensional vectors to be stored.

Algorithm 5.7.1 (Conjugate gradient Method)

1. **Initialization:** *Given a starting point* $\mathbf{x}^{(1)}$, *a tolerance* $\epsilon_{\mathbf{g}} > 0$ *and a maximum number of iterations* k_{max}, *set* $\beta^{(1)} = 0$ *and* $k = 1$.
2. *Calculate the objective value* $f^{(k)} = f(\mathbf{x}^{(k)})$ *and gradient* $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
3. *While* $k < k_{max}$ *and* $\|\mathbf{g}^{(k)}\| > \epsilon_{\mathbf{g}}$
   (a) **Search direction:**
      i. *If* $k = 1$ *then* $\mathbf{d}^{(1)} = -\mathbf{g}^{(1)}$ *(steepest descent direction).*
      ii. *If* $k > 1$ *calculate* $\beta^{(k+1)}$ *and* $\mathbf{d}^{(k)} = -\mathbf{g}^{(k)} + \beta^{(k)}\mathbf{d}^{(k-1)}$.
   (b) **Line search:** *Calculate an (exact or approximate) minimizer* $\alpha^{(k)}$ *of* $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$, *to find*
      i. **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$.
      ii. **New function value and gradient** $f^{(k+1)} = f(\mathbf{x}^{(k+1)})$, $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$.
   (c) *Increase iteration counter:* $k = k + 1$.
4. *End while loop*

The conjugate gradient method was first applied to function minimization by Fletcher and Reeves [FR64] in 1964, who used

$$\beta^{(k)} = \frac{{\mathbf{g}^{(k)}}^T \mathbf{g}^{(k)}}{{\mathbf{g}^{(k-1)}}^T \mathbf{g}^{(k-1)}} = \frac{\|\mathbf{g}^{(k)}\|_2^2}{\|\mathbf{g}^{(k-1)}\|_2^2}. \tag{5.7.1}$$

In 1971 Polak and Ribiere [PR69] proposed using

$$\beta^{(k)} = \frac{{\mathbf{g}^{(k)}}^T \left(\mathbf{g}^{(k)} - \mathbf{g}^{(k-1)}\right)}{{\mathbf{g}^{(k-1)}}^T \mathbf{g}^{(k-1)}} = \frac{{\mathbf{g}^{(k)}}^T \mathbf{y}^{(k-1)}}{\|\mathbf{g}^{(k-1)}\|_2^2}. \tag{5.7.2}$$

The parameter $\beta^{(k)}$ is chosen so that the conjugate gradient method when applied to a positive definite quadratic function using exact line searches will generate directions which are conjugate with respect to G. Proposition 5.4.4 then implies that the conjugate gradient methods will exhibit quadratic termination. These results depend on the fact that $\beta^{(1)} = 0$, so $\mathbf{d}^{(1)} = -\mathbf{g}^{(1)}$ the steepest descent direction. When applied to a strictly convex quadratic function using exact line searches the Fletcher–Reeves update (5.7.1) and the Polak–Ribiere update (5.7.2) are the same. Moreover in this case the conjugate gradient methods are identical to the quasi-Newton methods (see Fletcher [Fle87]).

Example 5.7.2 (Conjugate gradient method on a quadratic function)
Minimize $f(\mathbf{x}) = x_1^2 + 10x_2^2$ *starting from* $\mathbf{x}^{(1)} = [\ 1 \quad \frac{1}{10}\ ]^T$ *using a conjugate gradient method with the Polak-Ribiere update and exact line searches. Use exact arithmetic (`Maple` could help) and compare with Example 5.4.5.*

If a conjugate gradient method is used with approximate line searches, or it is used to minimize a non-quadratic function, then different choices for $\beta^{(k)}$ produce different iterates $\{\mathbf{x}^{(k)}\}$. It may also be necessary to reset the parameter $\beta^{(k)}$ to zero at certain intervals, for example $\beta^{(jn)} = 0$ for $j = 1, 2, \ldots$. If $\beta^{(k)} = 0$ then the steepest descent direction is use on the kth iteration. More details on the theoretical properties of conjugate gradient methods can be found in Fletcher [Fle87], while Gill, Murray and Wright [GMW81] discuss implementation issues.

Conjugate gradient methods require only $4n$ or $5n$ elements (plus scalars) to be stored. Moreover only $O(n)$ operations are required to calculate the search direction on each iteration and update the parameter $\beta^{(k)}$.

When A is a symmetric positive definite n by n matrix, solving the linear system

$$A\mathbf{x} = \mathbf{b}$$

is equivalent to minimizing the strictly convex quadratic function

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x},$$

as the gradient of $f(\mathbf{x})$ is the negative of the residual for the linear system

$$\mathbf{r}(\mathbf{x}) = \mathbf{b} - A\mathbf{x} = -\nabla f(\mathbf{x}).$$

In exact arithmetic and using an exact line search

$$\alpha^{(k)} = -\frac{\mathbf{d}^{(k)^T}\mathbf{g}^{(k)}}{\mathbf{d}^{(k)^T}A\mathbf{d}^{(k)}} = \frac{\mathbf{d}^{(k)^T}\mathbf{r}^{(k)}}{\mathbf{d}^{(k)^T}A\mathbf{d}^{(k)}},$$

the conjugate gradient method will find the minimizer in at most n iterations. For large n the conjugate gradient method is viewed as an iterative procedure in which a good estimate of the minimizer is obtained in many fewer than n iterations. In particular the components of the solution in the directions of the eigenvectors corresponding to the largest eigenvalues of A are found first. The conjugate gradient method exhibits linear convergence with rate

$$\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}$$

where

$$\kappa = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

is the spectral radius of A. If A has a large condition number then the rate of convergence can be prohibitively slow. Practical conjugate gradient methods require

- An efficient routine for calculating the matrix-vector product

  $$\mathbf{q} = A\mathbf{x}$$

  required to evaluate the residual (gradient). Typically A is very large and sparse, with only the non-zero elements of A being stored. In fact the matrix A itself does not need to be explicitly calculated as long as a routine for calculating $A\mathbf{x}$ for any $\mathbf{x}$ is provided.

- A symmetric positive definite preconditioner M to improve the rate of convergence. The conjugate gradient method is applied to the linear system $M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$, where M is chosen so that $\kappa_2(M^{-1}A) << \kappa_2(A)$. The other requirement for the preconditioner is that the linear system $M\mathbf{z} = \mathbf{r}$ must be much easier to solve than the original system $A\mathbf{x} = \mathbf{b}$. Preconditioners range from $M = \text{diag}(A_{11}, \ldots, A_{nn})$ to the incomplete Cholesky factorization of A (see Golub and van Loan [GV96] for example). The problem from which the linear system arose, for example as the discretization of a partial differential equation, can help provide a good preconditioner.

Algorithm 5.7.3 (Pre-conditioned conjugate gradient method)

1. **Initialization:** *Given a starting point* $\mathbf{x}^{(1)}$, *a tolerance* $\epsilon > 0$ *and a maximum number of iterations* $k_{max} \leq n$, *set* $\beta^{(1)} = 0$ *and* $k = 1$.
2. *Calculate* $\mathbf{r}^{(1)} = \mathbf{b} - A\mathbf{x}^{(1)}$.
3. *While* $k \leq k_{max}$ *and* $\|\mathbf{r}^{(k)}\| > \epsilon$
   (a) **Preconditioner:** *Solve* $M\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$.
   (b) $\rho^{(k)} = \mathbf{r}^{(k)^T}\mathbf{z}^{(k)}$.
   (c) **Search direction:** *If* $k = 1$ *then* $\mathbf{d}^{(k)} = \mathbf{z}^{(k)}$, *otherwise*

   $$\begin{aligned} \beta^{(k)} &= \rho^{(k)}/\rho^{(k-1)} \\ \mathbf{d}^{(k)} &= \mathbf{z}^{(k)} + \beta^{(k)}\mathbf{d}^{(k-1)} \end{aligned}$$

   (d) **Matrix-vector product:** $\mathbf{q}^{(k)} = A\mathbf{d}^{(k)}$.
   (e) **Line search:** $\alpha^{(k)} = \rho^{(k)}/\mathbf{d}^{(k)^T}\mathbf{q}^{(k)}$.

(f) **New point:** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$.

(g) **New residual:** $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha^{(k)}\mathbf{q}^{(k)}$.

(h) Increase iteration counter: $k = k + 1$.

4. End while loop

Important features of the conjugate gradient method are that (in exact arithmetic) it generates orthogonal residuals

$$\mathbf{r}^{(k+1}{}^{T}\mathbf{r}^{(k)} = 0$$

and directions which are conjugate with respect to A so

$$\mathbf{d}^{(k+1}{}^{T}A\mathbf{d}^{(k)} = 0.$$

Example 5.7.4 (Conjugate gradient on a linear system) *A central difference approximation of the the negative Laplacian*

$$-\frac{\partial^2 w(u,v)}{\partial u^2} - \frac{\partial^2 w(u,v)}{\partial v^2}$$

at equally space points produces a symmetric positive definite coefficient matrix A which is sparse and banded. An example of the sparse matrix arising from discretizing the Laplacian is given in Example A.2.1 on page 226 of Appendix A. The values of $\|\mathbf{r}^{(k)}\|$ and $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|$ for the conjugate gradient method without

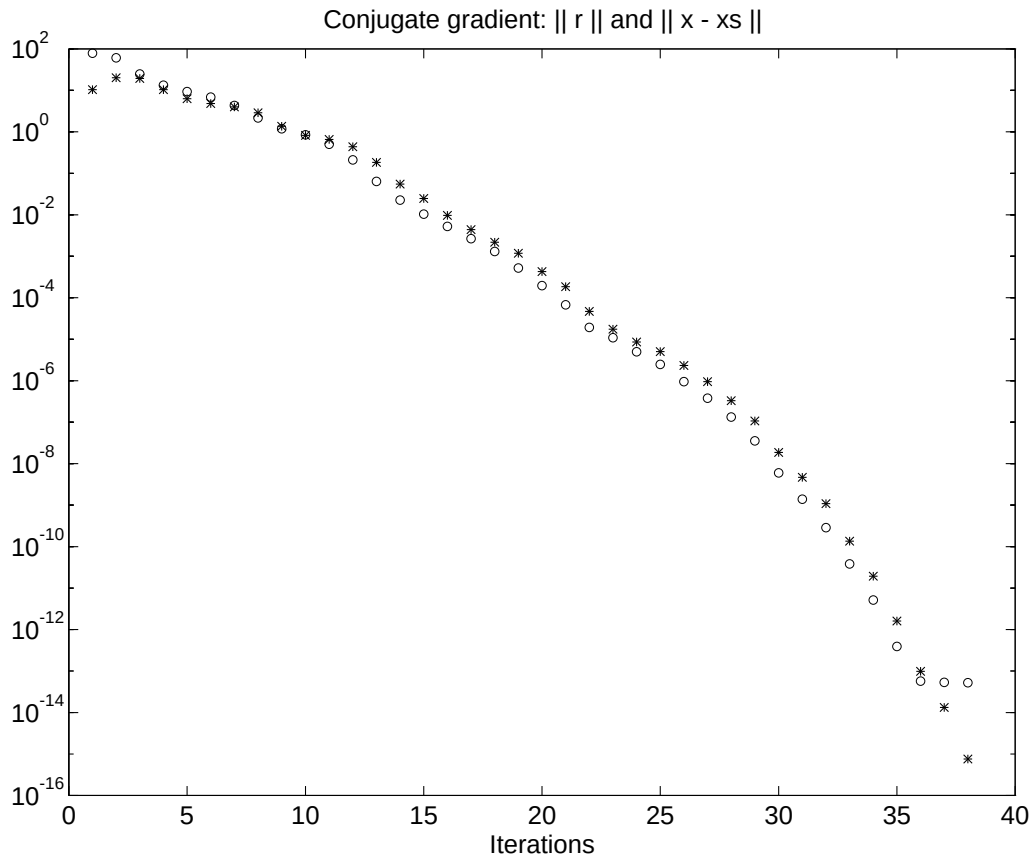


Figure 5.7.1: Conjugate gradient method on a linear system: $\|\mathbf{r}^{(k)}\|$ and $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|$

any preconditioning ($M = I$) are plotted in Figure 5.7.1. The convergence test $\|\mathbf{r}^{(k)}\| \leq 10^{-14}$ is satisfied

in 38 iterations, which is much less than the number of variables $n = 150$. The use of a preconditioner can reduce the number of iterations, but increases the cost of each iteration by the time taken to solve the linear system $M\mathbf{z} = \mathbf{r}$. The eigenvalues of A and $M^{-1}A$, for a preconditioner M based on an incomplete Cholesky factorization [GV96], are plotted in Figure 5.7.2. Clearly the eigenvalues of A are much closer

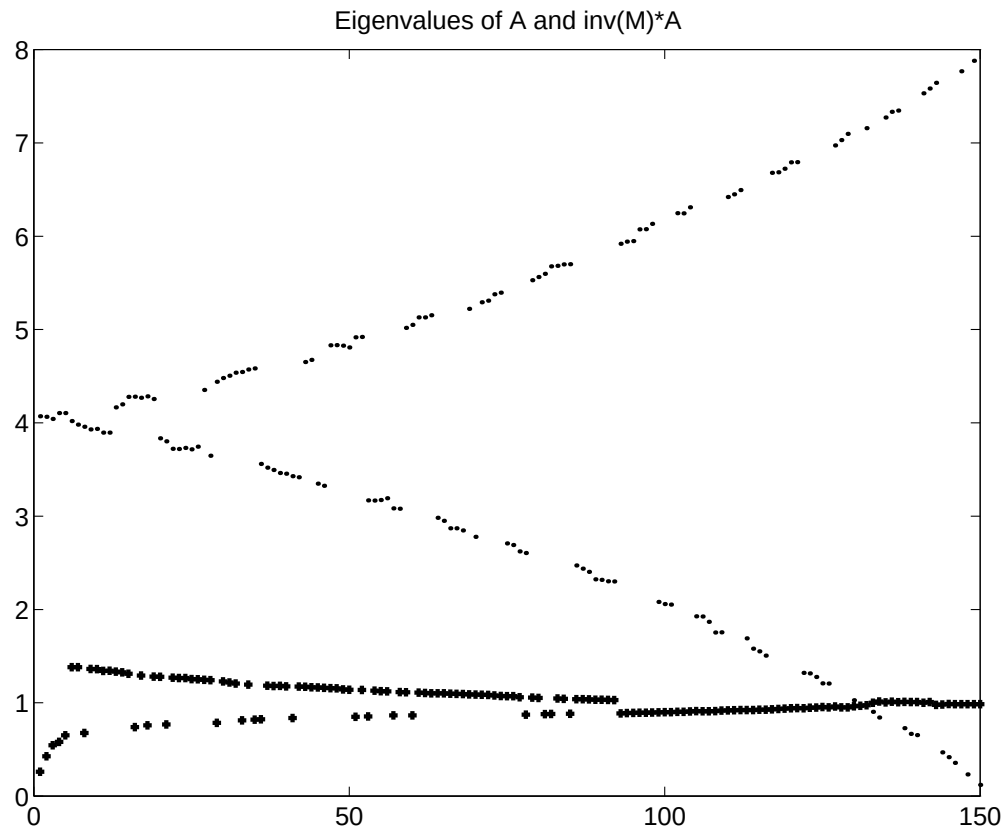


Figure 5.7.2: Eigenvalues of A and preconditioned system $M^{-1}A$

together. The condition number of A is 66, while the condition number of $M^{-1}A$ is 5.7. This example, including other iterative solution methods, is in the MATLAB *file* `pde.m`.

5.7.2 Newton methods using sparse matrix factorizations

When the non-zero elements of the Hessian can be evaluated it may be possible to solve the linear system $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$ using a sparse Cholesky factorization. A major problem in calculating the factorization of a sparse matrix is fill-in. Fill-in occurs when non-zeros elements are created in positions which were originally zero. This increase the storage requirements and the number of arithmetic operations. There are several methods of re-ordering the variables and/or equations with the aim of minimizing the amount of fill-in, and so making the Cholesky factorization as efficient as possible. If the matrix is banded, with bandwidth ω, so all the non-zero elements lie within a band about the main diagonal ($G_{ij}^{(k)} = 0$ for all $j - i| > \omega$) then fill-in can only occur within the band. Thus the reverse Cuthil-McKee re-ordering tries to reduce than bandwidth ω of the matrix. Other re-orderings are column count and minimum degree [GL89]. More details on direct methods for solving sparse linear systems can be found in George and Liu [GL81] and the SPARSPAK package [].

Add example of Hessian, full Cholesky, Cholesky after re-orderings.

5.7.3 Incomplete Newton methods

Iterative methods can be sued to solve the linear system $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$ to get a search direction. If $\mathbf{g}^{(k)}$ is positive definite then a pre-conditioned conjugate gradient method is typically used.

The accuracy with which the iterative method solves the linear system must be specified. The idea of the incomplete Newton methods [DES82] is that the overall efficiency of the solution process may be improved by only finding an approximate Newton direction when the iterate is far from a solution. As the iterates approach a solution the Newton direction should be calculated with higher accuracy. This requires a relation between the accuracy of the current iterate, for example as measured by $\|\mathbf{g}^{(k)}\|$, and the accuracy with which the iterate method finds the Newton direction.

5.7.4 Sparse quasi-Newton approximations

An attractive idea is to find a quasi-Newton approximation which keeps the known sparsity pattern of the Hessian. Except for partially separable problems, where the Hessian is block diagonal and each diagonal block can be approximated, this approach has not worked well.

5.7.5 Limited memory quasi-Newton methods

The key idea in limited memory quasi-Newton methods is to only keep the information from τ of the previous iterations, where typically $1 \leq \tau \leq 10$. This implies that the quasi-Newton approximation must be of low rank. The full Hessian approximation $B^{(k)}$ is never formed. Rather the information $S^{(k)} = \begin{bmatrix} \mathbf{s}^{(k)} & \cdots & \mathbf{s}^{(k-\tau)} \end{bmatrix}$ about previous steps and $Y^{(k)} = \begin{bmatrix} \mathbf{y}^{(k)} & \cdots & \mathbf{y}^{(k-\tau)} \end{bmatrix}$ about the previous gradient differences. This information is used directly to calculate the search direction. More details can be found in Nocedal [].

5.7.6 Exercises

1. Use the Fletcher-Reeves method (i.e. a conjugate gradient method with the Fletcher-Reeves update for $\beta^{(k)}$) to minimize
$$f(x) = x_1^2 - 2x_1x_2 + 5x_2^2 + 3x_1 - 12x_2 - 22$$
starting from $\mathbf{x}^{(1)} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and using exact line searches. Use exact arithmetic and compare your answers to those obtained in Exercise 1.

2. What are the storage requirements of the Fletcher–Reeves and Polak–Ribiere conjugate gradient methods?

3. [**H**] Prove that when applied to a strictly convex quadratic function with Hessian G a quasi-Newton method using an update from the Broyden family and exact line searches, terminates after $K \leq n$ iterations and for all $i \leq K$
$$\mathbf{s}^{(i)^T} G\mathbf{s}^{(j)} = 0 \quad \forall j < i,$$
$$B^{(i+1)}\mathbf{s}^{(j)} = \mathbf{y}^{(j)} \quad \forall j \leq i.$$
Hence show that if $K = n$ then $B^{(n+1)} = G$.

   Hint: Prove these two results inductively, then use the result that conjugate search directions plus exact line searches gives quadratic termination.

4. [**H**] Consider minimizing an arbitrary positive definite quadratic function with the following methods using exact line searches. Show that

   (a) the Polak-Ribiere and Fletcher-Reeves conjugate gradient methods generate the same sequence of points.

   (b) all members of the Broyden family of quasi-Newton methods with $\phi \in [0, 1]$ generate the same sequence of points.

   (c) the sequence of points generated by the conjugate gradient methods in (a) and the quasi-Newton methods in (b) are the same if $\beta^{(1)} = 0$ and $B^{(1)} = I$.

5. Run the `sparsity` demo in MATLAB to see an example of a sparse matrix A, the fill in that can occur when forming $A^T A$ and calculating the Choleski factorization of $A^T A$. It also demonstrates the effect of different re-orderings of the variables.

Chapter 6

Methods for Linearly Constrained Problems

6.1 Introduction

A linearly constrained optimization problem can be written as

$$\begin{array}{ll} Minimize & f(\mathbf{x}) \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & \mathbf{a}_i^T\mathbf{x} = b_i \quad i \in \mathcal{E} \\ & \mathbf{a}_i^T\mathbf{x} \leq b_i \quad i \in \mathcal{I}, \end{array} \tag{6.1.1}$$

where $\mathbf{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for $i \in \mathcal{E} \cup \mathcal{I}$. In standard form the feasible region is

$$\Omega = \{\, \mathbf{x} \in \mathbb{R}^n : c_i(\mathbf{x}) = 0 \; i \in \mathcal{E}, c_i(\mathbf{x}) \leq 0 \; i \in \mathcal{I} \,\}$$

where $c_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i$. Linear constraints include as special cases

- Simple lower bounds $\mathbf{x} \geq \mathbf{l}$, i.e $-x_i + l_i \leq 0$;
- Simple upper bounds $\mathbf{x} \leq \mathbf{u}$, i.e $x_i - u_i \leq 0$.

Exploiting the structure of simple lower and upper bounds is vital in practical numerical methods for solving linearly constrained optimization problems. The linear constraints may also be written as

$$\begin{bmatrix} A_{\mathcal{E}} \\ A_{\mathcal{I}} \end{bmatrix} \mathbf{x} \begin{array}{c} = \\ \leq \end{array} \begin{bmatrix} \mathbf{b}_{\mathcal{E}} \\ \mathbf{b}_{\mathcal{I}} \end{bmatrix}$$

where $A_{\mathcal{E}} = [\, \mathbf{a}_1 \quad \cdots \quad \mathbf{a}_{m_{\mathcal{E}}} \,]^T$, $A_{\mathcal{I}} = [\, \mathbf{a}_{m_{\mathcal{E}}+1} \quad \cdots \quad \mathbf{a}_m \,]^T$, $m_{\mathcal{E}} = |\mathcal{E}|$, and $m = |\mathcal{E}| + |\mathcal{I}|$.

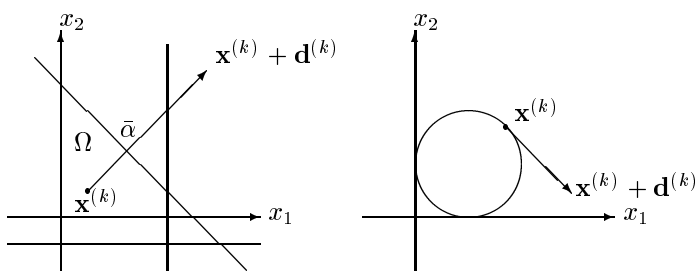


Figure 6.1.1: Feasible regions with linear and nonlinear constraints

Key properties of linearly constrained problems are

1. The constraints satisfy

$$\nabla c_i(\mathbf{x}) = \mathbf{a}_i, \quad \nabla^2 c_i(\mathbf{x}) = 0 \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}, \tag{6.1.2}$$

   so $\nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla^2 f(\mathbf{x})$.

2. If the feasible region Ω is non-empty, then it is a convex set, as all the constraints are affine.

3. If the feasible region Ω is not just a single point then at any feasible point $\mathbf{x}^{(k)} \in \Omega$ a search direction $\mathbf{d}^{(k)}$ and a step $\bar{\alpha} > 0$ can be found with

$$\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} \in \Omega \quad \text{for all } \alpha \in [0, \bar{\alpha}].$$

   Thus once a feasible point has been found, all subsequent iterates in a line search method can be forced to be feasible. This contrasts to the case of nonlinear constraints. as illustrated in Figure 6.1.1.

6.1.1 Finding a feasible point

The amount by which a constraint is violated (not satisfied) is

$$v_i = |c_i(\mathbf{x})| = |\mathbf{a}_i^T \mathbf{x} - \mathbf{b}_i| \quad \text{for } i \in \mathcal{E}, \tag{6.1.3a}$$

$$v_i = c_i^+(\mathbf{x}) \equiv \max(c_i(\mathbf{x}), 0) = \max(\mathbf{a}_i^T \mathbf{x} - b_i, 0) \quad \text{for } i \in \mathcal{I}. \tag{6.1.3b}$$

Let ν be the maximum constraint violation, so

$$\nu = \max_{i \in \mathcal{E} \cup \mathcal{I}} v_i \geq v_i \quad \text{for all } i \in \mathcal{E} \cup \mathcal{I}.$$

This amounts to taking the infinity norm $\|\mathbf{v}\|_\infty$ of the constraint violations (6.1.3a) and (6.1.3b). An inequality $\nu \geq |\mathbf{a}_i^T \mathbf{x} - b_i|$ can be converted into the pair of inequalities

$$\nu \geq |\mathbf{a}_i^T \mathbf{x} - b_i| \iff \nu \geq \mathbf{a}_i^T \mathbf{x} - b_i \text{ and } \nu \geq -(\mathbf{a}_i^T \mathbf{x} - b_i).$$

A feasible point can be found by solving

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n, \nu \in \mathbb{R}}{\textit{Minimize}} & \nu \\ \textit{Subject to} & \mathbf{a}_i^T \mathbf{x} - \nu \leq b_i \quad i \in \mathcal{E} \cup \mathcal{I} \\ & -\mathbf{a}_i^T \mathbf{x} - \nu \leq -b_i \quad i \in \mathcal{E} \\ & \nu \geq 0. \end{array} \tag{6.1.4}$$

The problem (6.1.4) is a linear programming problem in the $n+1$ variables $\mathbf{x}, \nu$. Given any $\mathbf{x}^{(1)} \in \mathbb{R}^n$ a feasible starting point for (6.1.4) is provided by taking

$$\nu^{(1)} = \max\left(\max_{i \in \mathcal{E}} |\mathbf{a}_i^T \mathbf{x}^{(1)} - b_i|, \ \max_{i \in \mathcal{I}} \mathbf{a}_i^T \mathbf{x}^{(1)} - b_i, \ 0 \right)$$

Thus an initial feasible point to an arbitrary linearly constrained problem can be found by solving a linear programming problem with a known initial feasible point.

An alternative is to minimize the 1-norm of the constraint violations.

$$\|\mathbf{v}\|_1 = \sum_{i \in \mathcal{E} \cup \mathcal{I}} |v_i| = \sum_{i \in \mathcal{E} \cup \mathcal{I}} v_i = \mathbf{e}^T \mathbf{v}$$

as each constraint violation $v_i \geq 0$ from (6.1.3). Here $\mathbf{e} \in \mathbb{R}^m$, where $m = |\mathcal{E}| + |\mathcal{I}|$, is a vector of all ones. This produces the linear programming problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n, \mathbf{v} \in \mathbb{R}^m}{\textit{Minimize}} & \sum_{i=1}^m v_i \\ \textit{Subject to} & \mathbf{a}_i^T \mathbf{x} - v_i \leq b_i \quad i \in \mathcal{E} \cup \mathcal{I} \\ & -\mathbf{a}_i^T \mathbf{x} - v_i \leq -b_i \quad i \in \mathcal{E} \\ & v_i \geq 0 \quad i \in \mathcal{E} \cup \mathcal{I}. \end{array} \tag{6.1.5}$$

This is a linear programming problem with $n+m$ variables $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^m$ and $m+|\mathcal{E}|$ general linear constraints and simple lower bounds $\mathbf{v} \geq 0$. For any value of $\mathbf{x}^{(1)}$ an initial feasible point for (6.1.5) is provided by

$$\begin{aligned} v_i^{(1)} &= |\mathbf{a}_i^T \mathbf{x}^{(1)} - b_i| \quad \text{for } i \in \mathcal{E}, \\ v_i^{(1)} &= \max\{\mathbf{a}_i^T \mathbf{x}^{(1)} - b_i, 0\} \quad \text{for } i \in \mathcal{I}. \end{aligned}$$

The problem (6.1.5) has more variables than (6.1.4), but has the advantage of trying to minimize all the constraint violations rather than just the largest constraint violation.

Simple lower and upper bounds can easily be dealt with explicitly. An initial feasible point that is close to a provided starting point $\mathbf{x}^{(1)}$ which does not satisfy the simple lower and upper bounds $\ell \leq \mathbf{x} \leq \mathbf{u}$ can be calculated by

$$\mathbf{x}_{feas}^{(1)} = \max\{\ell, \min\{\mathbf{x}^{(1)}, \mathbf{u}\}\}.$$

Typically the equality constraints are ordered as the first $m_{\mathcal{E}} = |\mathcal{E}|$ constraints of the matrix A, followed by the remaining $m - m_{\mathcal{E}}$ inequality constraints. Thus the general linear constraints are

$$\begin{bmatrix} A_{\mathcal{E}} \\ A_{\mathcal{I}} \end{bmatrix} \mathbf{x} \begin{matrix} = \\ \leq \end{matrix} \begin{bmatrix} \mathbf{b}_{\mathcal{E}} \\ \mathbf{b}_{\mathcal{I}} \end{bmatrix}.$$

In matrix notation (6.1.4) is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n, \nu \in \mathbb{R}}{Minimize} & \nu \\ Subject\ to & \begin{bmatrix} -A & \mathbf{e} \\ -A_{\mathcal{E}} & \mathbf{e}_{\mathcal{E}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \nu \end{bmatrix} \leq \begin{bmatrix} \mathbf{b} \\ -\mathbf{b}_{\mathcal{E}} \end{bmatrix} \\ & 0 \leq \nu. \end{array}$$

while (6.1.5) is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n, \mathbf{v} \in \mathbb{R}^m}{Minimize} & \mathbf{e}^T \mathbf{v} \\ Subject\ to & \begin{bmatrix} -A & I_m \\ -A_{\mathcal{E}} & \begin{bmatrix} I_{m_{\mathcal{E}}} & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix} \leq \begin{bmatrix} \mathbf{b} \\ -\mathbf{b}_{\mathcal{E}} \end{bmatrix} \\ & 0 \leq \mathbf{v}. \end{array}$$

6.1.2 Feasible directions

Let $\mathbf{x}^{(k)}$ be a feasible point, so $c_i(\mathbf{x}^{(k)}) = 0$ for $i \in \mathcal{E}$ and $c_i(\mathbf{x}^{(k)}) \leq 0$ for $i \in \mathcal{I}$. A direction $\mathbf{d}$ is feasible at $\mathbf{x}^{(k)}$ if there exists an $\bar{\alpha} > 0$ such that

$$\begin{aligned} c_i(\mathbf{x}^{(k)} + \alpha \mathbf{d}) &= 0 \quad i \in \mathcal{E} \\ c_i(\mathbf{x}^{(k)} + \alpha \mathbf{d}) &\leq 0 \quad i \in \mathcal{I} \end{aligned}$$

for all $\alpha \in [0, \bar{\alpha}]$. If $c_i(\mathbf{x}^{(k)}) < 0$ then this is true for any $\mathbf{d} \in \mathbb{R}^n$. Thus only constraints in the set $\mathcal{A}(\mathbf{x}^{(k)}) = \{i \in \mathcal{E} \cup \mathcal{I} : c_i(\mathbf{x}^{(k)}) = 0\}$ of active constraints at $\mathbf{x}^{(k)}$ restrict the choice of $\mathbf{d}$. Let $i \in \mathcal{A}(\mathbf{x}^{(k)}) \cap \mathcal{I}$, so $c_i(\mathbf{x}^{(k)}) = 0$. Then

$$c_i(\mathbf{x}^{(k)} + \alpha \mathbf{d}) \leq 0 \Longrightarrow c_i(\mathbf{x}^{(k)}) + \alpha \mathbf{a}_i^T \mathbf{d} \leq 0 \Longrightarrow \mathbf{a}_i^T \mathbf{d} \leq 0.$$

Similarly for $i \in \mathcal{E} \subseteq \mathcal{A}(\mathbf{x}^{(k)})$

$$c_i(\mathbf{x}^{(k)} + \alpha \mathbf{d}) = 0 \Longrightarrow c_i(\mathbf{x}^{(k)}) + \alpha \mathbf{a}_i^T \mathbf{d} = 0 \Longrightarrow \mathbf{a}_i^T \mathbf{d} = 0.$$

Thus at a feasible point $\mathbf{x}^{(k)}$ a direction $\mathbf{d}$ is feasible if

$$\begin{aligned} \mathbf{a}_i^T \mathbf{d} &= 0 \quad i \in \mathcal{E} \\ \mathbf{a}_i^T \mathbf{d} &\leq 0 \quad i \in \mathcal{A}(\mathbf{x}^{(k)}) \cap \mathcal{I}. \end{aligned}$$

6.1.3 Feasible line searches

To maintain feasibility of the new iterate $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$ the step length must satisfy

$$\begin{aligned} c_i(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) &= 0 \quad i \in \mathcal{E} \\ c_i(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) &\leq 0 \quad i \in \mathcal{I}. \end{aligned}$$

For a feasible direction these conditions are satisfied for any active constraints. Consider an inactive constraint, $i \notin \mathcal{A}(\mathbf{x}^{(k)})$, so $c_i(\mathbf{x}^{(k)}) < 0$. If $\mathbf{a}_i^T\mathbf{d}^{(k)} \leq 0$ then

$$c_i(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) = \mathbf{c}_i(\mathbf{x}^{(k)}) + \alpha\mathbf{a}_i^T\mathbf{d}^{(k)} < 0 \quad \forall\, \alpha \geq 0.$$

Thus only constraints for which $\mathbf{a}_i^T\mathbf{d}^{(k)} > 0$ limit the step that can be taken in the direction $\mathbf{d}^{(k)}$. If $\mathbf{a}_i^T\mathbf{d}^{(k)} > 0$ then α must satisfy

$$c_i(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) = \mathbf{c}_i(\mathbf{x}^{(k)}) + \alpha\mathbf{a}_i^T\mathbf{d}^{(k)} < 0 \Longrightarrow \alpha < \frac{-c_i(\mathbf{x}^{(k)})}{\mathbf{a}_i^T\mathbf{d}^{(k)}}.$$

As all constraints must be satisfied, a step $\alpha^{(k)}$ in a feasible direction $\mathbf{d}^{(k)}$ maintains feasibility if

$$\alpha^{(k)} \leq \bar{\alpha}^{(k)} \equiv \min_{\substack{i:\mathbf{a}_i^T\mathbf{d}^{(k)}>0 \\ c_i^{(k)}<0}} \frac{-c_i(\mathbf{x}^{(k)})}{\mathbf{a}_i^T\mathbf{d}^{(k)}}. \tag{6.1.6}$$

6.1.4 Active set methods

Let $\mathcal{M}^{(k)} \subseteq \mathcal{A}(\mathbf{x}^{(k)})$ be an approximation to $\mathcal{A}(\mathbf{x}^*)$. Usually $\mathcal{M}^{(k)}$ will also have the property that the constraint gradients $\mathbf{a}_i$ for $i \in \mathcal{M}^{(k)}$ are linearly independent.

Algorithm 6.1.1 (Active set method for linear constraints) *Let $\mathbf{x}^{(1)}$ be a given initial feasible point. Let $\mathcal{M}^{(1)} \subseteq \mathcal{A}(\mathbf{x}^{(1)})$ such that $\mathbf{a}_i$ $i \in \mathcal{M}^{(1)}$ are linearly independent be a given initial active set approximation. Let $k = 1$.*

1. *Let $t^{(k)} = |\mathcal{M}^{(k)}|$.*
   (a) *If $t^{(k)} = n$ then $\mathbf{g}_Z^{(k)} = 0$.*
   (b) *If $t^{(k)} < n$ then*
      i. *Calculate an (orthogonal) matrix $Z^{(k)} \in \mathbb{R}^{n\times n-t^{(k)}}$ of full column rank such that*
      $$(Z^{(k)})^T\mathbf{a}_i = 0 \quad \forall\, i \in \mathcal{M}^{(k)}.$$
      ii. *Calculate the reduced gradient $\mathbf{g}_Z^{(k)} = (Z^{(k)})^T\nabla f(\mathbf{x}^{(k)})$.*
2. **Check optimality**
   If $\mathbf{g}_Z^{(k)} = 0$
   (a) *Solve the linear system*
   $$\sum_{i\in\mathcal{M}^{(k)}} \mathbf{a}_i\lambda_i + \nabla f(\mathbf{x}^{(k)}) = 0$$
   for the Lagrange multipliers $\lambda_i^{(k)}$, $i \in \mathcal{M}^{(k)}$.
   (b) *Check the sign of the multipliers corresponding to active inequality constraints. Let*
   $$\lambda_j^{(k)} = \min_{i\in\mathcal{M}^{(k)}\cap\mathcal{I}} \lambda_i^{(k)}.$$
      i. *If $\lambda_j^{(k)} \geq 0$ then* STOP *as $\mathbf{x}^{(k)}$ satisfies the necessary conditions for a local minimizer.*

ii. If $\lambda_j^{(k)} < 0$ *then drop the index* j *from the set* $\mathcal{M}^{(k)}$

$$\mathcal{M}_+^{(k)} = \mathcal{M}^{(k)} - \{j\}.$$

If $\mathbf{g}_Z^{(k)} \neq 0$ *then set* $\mathcal{M}_+^{(k)} = \mathcal{M}^{(k)}$.

3. **Equality constrained subproblem**

Solve (approximately) the equality constrained subproblem

$$\begin{array}{ll} Minimize & f(\mathbf{x}^{(k)} + \mathbf{d}) \\ \mathbf{d} \in \mathbb{R}^n & \\ Subject\ to & \mathbf{a}_i^T \mathbf{d} = 0 \quad i \in \mathcal{M}_+^{(k)}. \end{array} \tag{6.1.7}$$

for a search direction $\mathbf{d}^{(k)}$ *with* ${\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)}) < 0$.

4. **Constrained line search**

   (a) *Find the distance* $\bar{\alpha}^{(k)}$ *to the first constraint to become active.*

$$\bar{\alpha}^{(k)} = \begin{cases} \infty & \text{if } \mathbf{a}_i^T \mathbf{d}^{(k)} \leq 0 \text{ for all } i, \\ \min_{i:\mathbf{a}_i^T \mathbf{d}^{(k)} > 0} \frac{-c_i^{(k)}}{\mathbf{a}_i^T \mathbf{d}^{(k)}} & \text{otherwise} \end{cases}$$

   If $\bar{\alpha}^{(k)} < \infty$ *let* l *be an index which achieves the minimum, so*

$$\bar{\alpha}^{(k)} = \frac{-c_l^{(k)}}{\mathbf{a}_l^T \mathbf{d}^{(k)}}.$$

   (b) *Calculate an exact or approximate constrained minimizer* $\alpha^{(k)} \leq \bar{\alpha}^{(k)}$:

$$\alpha^{(k)} = \underset{\alpha \leq \bar{\alpha}^{(k)}}{argmin}\ f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}). \tag{6.1.8}$$

   (c) *Update the active set approximation. Set*

$$\mathcal{M}^{(k+1)} = \begin{cases} \mathcal{M}_+^{(k)} & \text{if } \alpha^{(k)} < \bar{\alpha}^{(k)} \\ \mathcal{M}_+^{(k)} \cup \{l\} & \text{if } \alpha^{(k)} = \bar{\alpha}^{(k)} \end{cases}$$

5. *Set* $k = k + 1$ *and go to Step 1.*

6.1.5 Equality constrained subproblems

Instead of solving the equality constrained subproblem (6.1.7) exactly a quadratic model of the objective $f(\mathbf{x}^{(k)} + \mathbf{d})$ can be made. This produces the equality constrained quadratic programming subproblem

$$\begin{array}{ll} Minimize & q_k(\mathbf{d}) \equiv f^{(k)} + \mathbf{d}^T \mathbf{g}^{(k)} + \frac{1}{2} \mathbf{d}^T H^{(k)} \mathbf{d} \\ \mathbf{d} \in \mathbb{R}^n & \\ Subject\ to & \mathbf{a}_i^T \mathbf{d} = 0 \quad i \in \mathcal{M}^{(k)}. \end{array} \tag{6.1.9}$$

Let $Z^{(k)} \in \mathbb{R}^{n \times n - t^{(k)}}$ be a basis for the tangent space to the constraints in $\mathcal{M}^{(k)}$, so $Z^{(k)}$ has full rank and $\mathbf{a}_i^T Z^{(k)} = 0$ for all $i \in \mathcal{M}^{(k)}$. Numerically an orthogonal matrix $Z^{(k)}$, so that $(Z^{(k)})^T Z^{(k)} = I$, is preferable. Define

1. The *reduced gradient* $\mathbf{g}_Z^{(k)} = (Z^{(k)})^T \mathbf{g}^{(k)}$.

2. The *reduced Hessian* $H_Z^{(k)} = (Z^{(k)})^T H^{(k)} Z^{(k)}$.

Then any feasible $\mathbf{d}$ for (6.1.9) can be expressed as

$$\mathbf{d} = Z^{(k)}\mathbf{v}, \quad \mathbf{v} \in \mathbb{R}^{n-t^{(k)}}$$

so the equality constrained problem (6.1.9) reduces to the unconstrained problem

$$\begin{array}{ll} Minimize & \mathbf{v}^T\mathbf{g}_Z^{(k)} + \frac{1}{2}\mathbf{v}^T H_Z^{(k)}\mathbf{v}. \\ v \in \mathbb{R}^{n-t^{(k)}} & \end{array} \tag{6.1.10}$$

If $H_Z^{(k)}$ is positive definite the solution of (6.1.9) is

$$\mathbf{d}^{(k)} = Z^{(k)}\mathbf{d}_Z^{(k)} \quad \text{where } H_Z^{(k)}\mathbf{d}_Z^{(k)} = -\mathbf{g}_Z^{(k)}.$$

Here $\mathbf{d}_Z^{(k)}$ is the solution of (6.1.10).

The quadratic programming subproblem has a unique minimizer if the reduced Hessian $H_Z^{(k)}$ is positive definite. This is automatically true of $H^{(k)}$ is positive definite. The Hessian $H^{(k)}$ of the quadratic model can be the identity matrix, the Hessian $G^{(k)}$ of the objective or an approximation to $G^{(k)}$. If the objective $f(\mathbf{x})$ is quadratic then a quadratic model of f obtained from a second order Taylor series will be exact. In this case using $H^{(k)} = \nabla^2 f(\mathbf{x}^{(k)})$ means that the solution to the quadratic subproblem (6.1.9) is the exact minimizer of (6.1.7).

The reduced gradient $\mathbf{g}_Z^{(k)} = 0$ if and only if there exist multipliers $\lambda_i^{(k)}$ for $i \in \mathcal{M}^{(k)}$ such that

$$\sum_{i \in \mathcal{M}^{(k)}} \lambda_i^{(k)}\mathbf{a}_i + \mathbf{g}^{(k)} = 0,$$

so $\mathbf{x}^{(k)}$ is a constrained stationary point.

Example 6.1.2 *Solve the problem*

$$\begin{array}{ll} \min & (x_1+2)^2(x_2^4+1) \\ \mathbf{x} \in \mathbb{R}^2 & \\ \textit{subject to} & x_1 + 2x_2 - 2 \le 0 \\ & x_1 - 2x_2 - 2 \le 0 \\ & x_1 - 1 \le 0, \end{array}$$

using a single projected Newton step at each iteration, rather than solving the equality constrained subproblem exactly. Start at the point $\mathbf{x}^{(1)} = [\,2 \quad 0\,]^T$ *with all constraints active, so* $\mathcal{M}^{(k)} = \mathcal{A}(\mathbf{x}^{(k)})$. *The constraints (dashed lines) and objective contours are drawn in Figure 6.1.2.*

6.1.6 Exercises

1. The linear program

$$\begin{array}{ll} Minimize & \nu \\ \mathbf{x} \in \mathbb{R}^n, \nu \in \mathbb{R} & \\ Subject\ to & \mathbf{a}_i^T\mathbf{x} - \nu \le b_i \quad i \in \mathcal{E} \cup \mathcal{I} \\ & -\mathbf{a}_i^T\mathbf{x} - \nu \le -b_i \quad i \in \mathcal{E} \\ & \nu \ge 0. \end{array} \tag{6.1.11}$$

to find a feasible point to systems of linear equations and inequalities comes from minimizing the infinity norm of the constraint violations v_i defined by

$$v_i = |c_i(\mathbf{x})| = |\mathbf{a}_i^T\mathbf{x} - \mathbf{b}_i| \quad \text{for } i \in \mathcal{E}, \tag{6.1.12a}$$

$$v_i = c_i^+(\mathbf{x}) \equiv \max(c_i(\mathbf{x}), 0) = \max(\mathbf{a}_i^T\mathbf{x} - b_i, 0) \quad \text{for } i \in \mathcal{I}. \tag{6.1.12b}$$

Give a linear program to find a feasible point to systems of linear equations and inequalities which is based on minimizing the 1-norm of the constraint violations. Show that for any $\mathbf{x}^{(1)}$ an initial feasible point can easily be obtained.

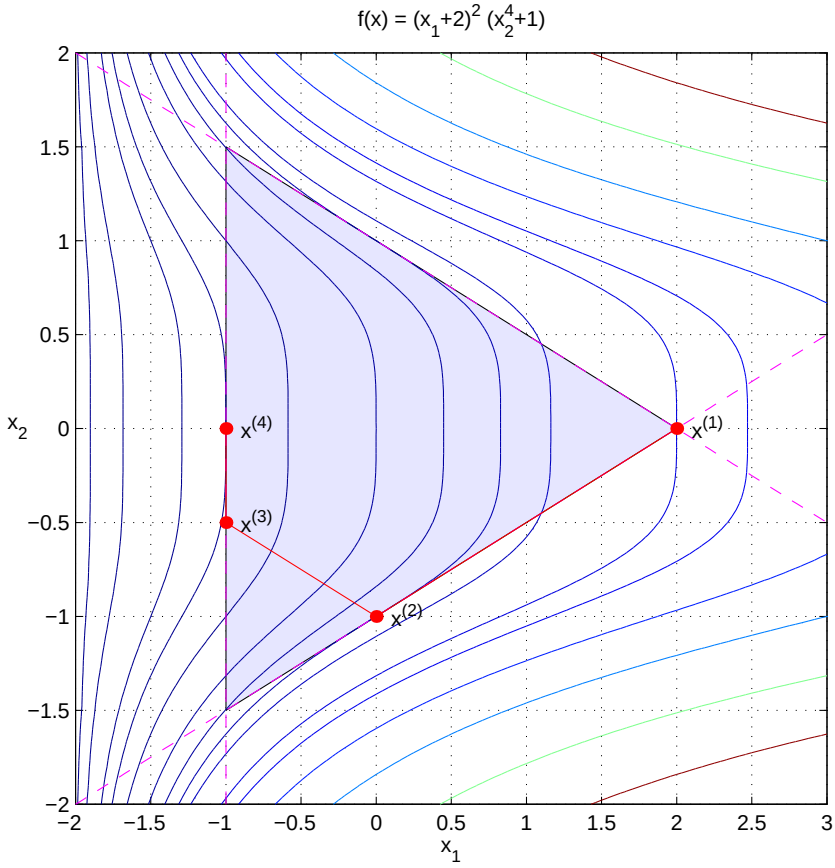


Figure 6.1.2: Constraints and objective contours for Example 6.1.2

6.2 An active set method for linear programming

As the objective is linear $f(\mathbf{x}^{(k)} + \mathbf{d}) = f(\mathbf{x}^{(k)}) + \mathbf{g}^T\mathbf{d}$, the equality constrained subproblem (6.1.7) does not have a finite minimum unless the constraints imply the only feasible point is $\mathbf{d} = 0$. Thus instead of solving (6.1.7) the search direction used is the steepest feasible descent direction

$$\mathbf{d}^{(k)} = -Z^{(k)}\mathbf{g}_Z^{(k)} = -Z^{(k)}{Z^{(k)}}^T\mathbf{g}^{(k)}.$$

If $\mathbf{x}^{(k)}$ is a vertex with $|\mathcal{M}^{(k)}| = n$ and $\mathcal{M}_+^{(k)} = n-1$ then $\mathbf{d}^{(k)}$ is a scalar multiple of the direction ${A^{(k)}}^{-1}\mathbf{e}_{\hat{\jmath}}$ where $\hat{\jmath}$ gives the column of $A^{(k)}$ corresponding to $\mathbf{a}_j$ (see Question 2).

The objective can be reduced by taking larger and larger steps along $\mathbf{d}^{(k)}$ until the constraint boundary is reached. If $\mathbf{a}_i^T\mathbf{d}^{(k)} \le 0$ for all $i \notin \mathcal{M}^{(k)}$ then the linear programming problem does not have a finite minimum. Otherwise the step is always the distance

$$\alpha^{(k)} = \frac{-c_l^{(k)}}{\mathbf{a}_l^T\mathbf{d}^{(k)}} = \min_{i:\mathbf{a}_i^T\mathbf{d}^{(k)}>0} \frac{-c_i^{(k)}}{\mathbf{a}_i^T\mathbf{d}^{(k)}}$$

to the first constraint to become active along the ray $\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}$, $\alpha \ge 0$. Thus for a linear programming problem the constrained line search (6.1.8) is not needed.

Example 6.2.1 *Solve the linear programming problem*

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^2}{Minimize} & f(\mathbf{x}) = 4x_1 - 5x_2 + 2 \\ Subject\ to & c_1(\mathbf{x}) = x_1 + x_2 - 3 \le 0 \\ & c_2(\mathbf{x}) = -x_1 - 1 \le 0 \\ & c_3(\mathbf{x}) = x_1 - 2 \le 0 \\ & c_4(\mathbf{x}) = -x_2 \le 0 \end{array} \tag{6.2.1}$$

using an active set method starting from $\mathbf{x}^{(1)} = [\ 0 \quad 0\ ]^T$.

The feasible region is sketched in Figure 6.2.1. As all the constraints are inequalities $\mathcal{E} = \emptyset$ *and*

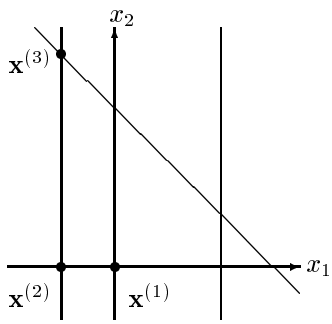


Figure 6.2.1: Linear programming example

$\mathcal{I} = \{1, \ldots, 5\}$. *The gradients of the objective and constraints functions are*

$$\mathbf{g} = \begin{bmatrix} 4 \\ -5 \end{bmatrix},\ \mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},\ \mathbf{a}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix},\ \mathbf{a}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ \mathbf{a}_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

At $\mathbf{x}^{(1)}$ *the constraint values are* $\mathbf{c}^{(1)} = [\ -3 \quad -1 \quad -2 \quad 0\ ]^T \le 0^T$, *so* $\mathbf{x}^{(1)}$ *is feasible, and the set of active constraints at* $\mathbf{x}^{(1)}$ *is* $\mathcal{A}^{(1)} = \{4\}$. *Taking* $\mathcal{M}^{(1)} = \mathcal{A}^{(1)}$ *gives* $A^{(1)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$. *Let*

$$Z^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Longrightarrow \mathbf{g}_Z^{(1)} = (Z^{(1)})^T\mathbf{g} = 4.$$

As the reduced gradient is non-zero this immediately gives the search direction

$$\mathbf{d}^{(1)} = -Z^{(1)}\mathbf{g}_Z^{(1)} = \begin{bmatrix} -4 \\ 0 \end{bmatrix}.$$

Then

$c_i^{(1)}$	-3	-1	-2	0
$\mathbf{a}_i^T\mathbf{d}^{(1)}$	-4	4	-4	0
$\frac{-c_i^{(1)}}{\mathbf{a}_i^T\mathbf{d}^{(1)}} : \mathbf{a}_i^T\mathbf{d}^{(1)} > 0$		$\frac{1}{4}$		

The next iterate is

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha^{(1)}\mathbf{d}^{(1)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} -4 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

where $\mathcal{M}^{(2)} = \{2, 4\}$.

At $\mathbf{x}^{(2)}$ *the objective value is* $f(\mathbf{x}^{(2)}) = -2 < f(\mathbf{x}^{(1)}) = 2$, *so the objective value has decreased as expected. It is also worth checking that* $\mathbf{c}^{(2)} = [\ -4 \quad 0 \quad -3 \quad 0\ ]^T \le 0$, *so* $\mathbf{x}^{(2)}$ *is feasible.* $A^{(2)} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$. *As* $\mathbf{x}^{(2)}$ *is a vertex of the feasible region* ($|\mathcal{M}^{(2)}| = n = 2$ *and the constraint gradients are*

linearly independent, so $A^{(2)}$ is an n by n nonsingular matrix), the reduced gradient is automatically zero. Solving $A^{(2)}\lambda^{(2)} + \mathbf{g} = 0$ gives $\lambda_2^{(2)} = 4$ and $\lambda_4^{(2)} = -5$. Then

$$\lambda_j = \min_{i \in \mathcal{M}^{(2)} \cap \mathcal{I}} \lambda_i^{(2)} = -5, \quad \textit{with} \quad j = 4.$$

As $\lambda_j < 0$ the point $\mathbf{x}^{(2)}$ is not a minimizer. Removing $j = 4$ from the active set gives $\mathcal{M}_+^{(2)} = \{2\}$ and hence $Z^{(2)} = [\,0 \quad 1\,]^T$. Thus

$$\mathbf{d}^{(2)} = -Z^{(2)}{Z^{(2)}}^T\mathbf{g} = -\begin{bmatrix} 0 \\ 1 \end{bmatrix}(-5) = \begin{bmatrix} 0 \\ 5 \end{bmatrix}.$$

Using this search direction

$c_i^{(2)}$	-4	0	-3	0
$\mathbf{a}_i^T\mathbf{d}^{(2)}$	5	0	0	-5
$\frac{-c_i^{(2)}}{\mathbf{a}_i^T\mathbf{d}^{(2)}} : \mathbf{a}_i^T\mathbf{d}^{(2)} > 0$	$\frac{4}{5}$			

The step to the nearest constraint is

$$\alpha^{(2)} = \min_{\mathbf{a}_i^T\mathbf{d}^{(2)} > 0} \frac{-c_i^{(2)}}{\mathbf{a}_i^T\mathbf{d}^{(2)}} = \frac{4}{5},$$

giving

$$\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha^{(2)}\mathbf{d}^{(2)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} + \frac{4}{5}\begin{bmatrix} 0 \\ 5 \end{bmatrix} = \begin{bmatrix} -1 \\ 4 \end{bmatrix}.$$

The active set approximation is $\mathcal{M}^{(3)} = \{1, 2\}$ and $A^{(3)} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}$. As $\mathbf{x}^{(3)}$ is a vertex the reduced gradient is automatically zero. Solving for the Lagrange multipliers

$$A^{(3)}\lambda^{(3)} + \mathbf{g} = 0 \Longrightarrow \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} + \begin{bmatrix} 4 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Longrightarrow \lambda_1^{(3)} = 5,\ \lambda_2^{(3)} = 9.$$

Hence

$$\lambda_j^{(3)} = \min_{i \in \mathcal{M}^{(3)} \cap \mathcal{I}} \lambda_i^{(3)} = 5 \geq 0$$

so $\mathbf{x}^{(3)} = [\,-1 \quad 4\,]^T$ is the minimizer with $f(\mathbf{x}^{(3)}) = -22$. Also $\mathbf{c}^{(3)} = [\,0 \quad 0 \quad -1 \quad -4\,]^T \leq 0$ confirming that $\mathbf{x}^{(3)}$ is feasible.

6.2.1 Exercises

1. Consider the linear programming problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & f(\mathbf{x}) = -2x_1 - x_2 \\ Subject\ to & c_1(\mathbf{x}) = x_1 + x_2 - 5 \leq 0 \\ & c_2(\mathbf{x}) = -x_1 + x_2 - 5 \leq 0 \\ & c_3(\mathbf{x}) = -x_1 - 2 \leq 0 \\ & c_4(\mathbf{x}) = x_1 - 4 \leq 0 \\ & c_5(\mathbf{x}) = -x_2 - 1 \leq 0 \end{array}$$

   (a) Sketch the feasible region for this problem.

   (b) Use the active set method for linear programming to solve this problem starting from

      i. $\mathbf{x}^{(1)} = [\,-2 \quad -1\,]^T$ (a vertex of the feasible region).

      ii. $\overline{\mathbf{x}}^{(1)} = [\,1 \quad 1\,]^T$ (a point in the interior of the feasible region).

      iii. $[\mathbf{H}]\hat{\mathbf{x}}^{(1)} = [\,5 \quad 5\,]^T$ (an infeasible point).

2. This question refers to the various steps of the active set method for the linear programming applied to the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & f(\mathbf{x}) = \mathbf{g}^T\mathbf{x} + f_0 \\ Subject\ to & c_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i = 0 \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i \leq 0 \quad i \in \mathcal{I} \end{array}$$

(a) Let $\mathbf{x}^{(k)}$ be a regular vertex of the feasible region Ω, and let $\lambda^{(k)}$ be the corresponding Lagrange multipliers. Let

$$\lambda_j^{(k)} = \min_{i \in \mathcal{A}(\mathbf{x}^{(k)}) \cap \mathcal{I}} \lambda_i^{(k)},$$

where $\mathcal{A}(\mathbf{x}^{(k)}) = \{\, i \in \mathcal{E} \cup \mathcal{I} : c_i(\mathbf{x}^{(k)}) = 0 \,\}$ is the set of active constraints at $\mathbf{x}^{(k)}$. Remember that a regular vertex of the feasible region Ω is a point $\mathbf{x} \in \Omega$ where $\mathcal{A}(\mathbf{x})$ has n elements and the gradients of the active constraints are linearly independent, so $A^{(k)} = [\,\mathbf{a}_i \text{ for } i \in \mathcal{A}(\mathbf{x}^{(k)})\,]$ is an $n \times n$ nonsingular matrix.

Show that if $\lambda_j^{(k)} < 0$ then $\mathbf{d}^{(k)} = -A^{(k)^{-T}}\mathbf{e}_{\hat{\jmath}}$ is a feasible descent direction at $\mathbf{x}^{(k)}$. Here $\hat{\jmath}$ is the position of the jth constraint in the set $\mathcal{A}(\mathbf{x}^{(k)})$, and $\mathbf{e}_{\hat{\jmath}}$ is the $\hat{\jmath}$th unit vector in $\mathbb{R}^n$.

(b) Let $\mathbf{d}^{(k)}$ be a feasible descent direction at a feasible point $\mathbf{x}^{(k)}$. Show that the minimizer $\alpha^{(k)}$ of $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$ on the feasible region is given by

$$\alpha^{(k)} = \min\left(\frac{-c_i^{(k)}}{\mathbf{a}_i^T\mathbf{d}^{(k)}} \quad \text{for } i : \mathbf{a}_i^T\mathbf{d}^{(k)} > 0\right).$$

(c) [**H**]If $\mathbf{x}$ is a vertex of the feasible region Ω then a direction $\mathbf{d} \in \mathbb{R}^n$ lies along an edge of Ω if there exists an $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha})$ the set of active constraints $\mathcal{A}(\mathbf{x} + \alpha\mathbf{d})$ has $n-1$ elements, and the gradients of the constraints in $\mathcal{A}(\mathbf{x} + \alpha\mathbf{d})$ are linearly independent.

Let $\mathbf{x}^{(k)}$ be regular vertex of Ω. Show that the directions

$$\mathbf{d}_j^{(k)} = A^{(k)^{-T}}\mathbf{e}_j$$

lie along the edges of the feasible region, and that $\mathbf{g}^T\mathbf{d}_j^{(k)} = \lambda_j^{(k)}$. Thus choosing $\mathbf{d}^{(k)}$ as in part (a) provides the steepest edge descent direction.

(d) [**H**]Let $\mathbf{x}^{(k)}$ be a regular vertex of the feasible region Ω and let $p \in \{1, \ldots, n\}$. Show that the search directions

$$\hat{\mathbf{d}} = A^{(k)^{-T}}\mathbf{e}_p \quad \text{and} \quad \mathbf{d} = -Z^{(k)}(Z^{(k)})^T\mathbf{g}$$

are equivalent in the sense that there exists a positive scalar η such that $\hat{\mathbf{d}} = \eta\mathbf{d}$. Here $Z^{(k)}$ is an $n \times (n-1)$ matrix of full rank such that $(Z^{(k)})^T\mathbf{a}_i = 0$ for all $i \in \mathcal{A}(\mathbf{x}^{(k)}) - \{p\}$.

3. Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Show that the KKT conditions for the problems

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{\min} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array}$$

and

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^m}{\max} & \lambda^T\mathbf{b} \\ \text{subject to} & A^T\lambda \geq \mathbf{c} \\ & \lambda \geq 0 \end{array}$$

are the same.

4. A firm has to schedule the monthly production of a certain item over a six month period. The production cost per item is \$10 in the first month, and subsequently rises at a rate of \$2 per month. Due to machinery changes during the final month there is an additional cost of \$4 for each item produced in the final month. The firm must meet monthly orders of 300, 600, 800, 800, 600, and 600 units. With normal shifts the firm can produce a maximum of 600 units per month. With the exception of the first and last months overtime can be used to increase production. Using overtime an extra 200 units per month can be produced, but at an increase of 50% in the production cost of each item. Excess production can be stored at a cost of \$3 per item for the first month of storage, and \$2 per item per month of storage thereafter.

   How should the firm schedule production to minimize the cost?

   (a) Formulate this problem as a linear programming problem.
   Hint: To facilitate formulation of the problem introduce variables u_{ij} and v_{ij} as follows:
   - u_{ij} = number of items produced, using normal shifts, in month i and sold in month j.
   - v_{ij} = number of items produced, using overtime shifts, in month i and sold in month j.

   (b) Solve the linear programming problem, ignoring any integer constraints on the variables, using the linear programming routine `lp` in MATLAB on the SIGMA. Interpret the results of this package. You must explain to the manager of the firm (who is not a mathematician) what his optimal schedule is, what it will cost, and whether it is unique.

5. A brewery blends and processes four components to produce three beers: draught, old fashioned and light. In making draught beer 5% of the volume of the components is lost in processing, while in making old fashioned 15% of the volume of the components is lost, and in making light 10% of the volume of the components is lost in processing. The quantities of each component that are available each week and the cost per barrel of each component are given in Table 6.2.1. There are a

Component	Barrels available per week	Cost per barrel
1	500	\$90
2	240	\$70
3	400	\$120
4	150	\$60

Table 6.2.1: Brewery data

number of restrictions governing the percentage of the components used to make the different beers. These percentages are measured as the volume of the component used divided by the volume of beer produced. Draught beer must have at least 40% of component 1, at most 20% of component 2, and at least 30% of component 3. Old fashioned beer must have at least 40% of component 3. Light beer must have at least 50% of component 2, and not more than 10% of component 1.

At least 200 barrels of light beer must be produced each week, while at most 500 barrels of old can be produced each week. The brewery sells draught beer for \$120 per barrel, old fashioned beer for \$180 per barrel and light beer for \$100 per barrel.

What is the optimal mix of the four components that will maximize the brewery's weekly profit?

   (a) Pose this as a mathematical optimization problem *in standard form.*
   Hint: It is not good enough just to let x_1 = the number of barrels of draught produced per week as you need to know how much of each component is in each barrel of draught. Using additional variables, which can be related to other variables through equality constraints, can simplify the formulation.

   (b) Say as much as you can about the structure of this problem.

   (c) Use one of the optimization packages available in MATLAB to solve this problem. Documentation is available by typing `help optim` in MATLAB.
   Explain your solution to the brewery manager, who is NOT a mathematician.

(d) Which constraints are active at the solution and which are not active?

(e) While presenting the report you are asked to estimate the change in profit if

i. The weekly minimum production of light beer increases by 10%?

ii. The availability of component 3 increases by 10%?

Explain how you could do this without resolving the problem.

6. The Transportation Department has to arrange for the delivery of *Gismos* from three supply depots at Bankstown, Botany and Hornsby to four factories at Wollongong, Mascot, West Ryde and Newcastle. They have estimated the cost in \$ per *Gismo* for transportation from each supply depot to each factory. They also have information on the number of *Gismos* available at each supply depot and the demand at each factory. The manager has decreed that all factories demands will be satisfied.

	Wollongong	West Ryde	Mascot	Newcastle	Supply
Bankstown	9	14	12	17	200
Botany	11	10	6	10	200
Hornsby	12	8	15	7	200
Demand	130	170	100	150	

Table 6.2.2: Transportation costs, supply and demand

Due to a union dispute **exactly** 20 *Gismos* are to be shipped from Botany to Mascot.

(a) How should the *Gismos* be shipped to minimize the total transportation costs, while meeting all factory demands.

(b) Without resolving the problem can you estimate the change in transportation cost if the union would agree to ship 30 *Gismos* from Botany to Mascot.

(c) The union may impose a work to rule which limits the amount that can be shipped from each supply depot to each factory. These limits are give in Table 6.2.3. Will the manager still be able to meet all factory demands and what will it cost if they can?

	Wollongong	West Ryde	Mascot	Newcastle
Bankstown	50	50	30	40
Botany	60	80	80	50
Hornsby	30	50	30	80

Table 6.2.3: Maximum number of *Gismos* that can be transported

6.3 Active set method for quadratic programming

A quadratic programming problem has a quadratic objective and linear constraints. A quadratic objective can be written as

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{g}_0^T\mathbf{x} + f_0$$

where $G \in \mathbb{R}^{n\times n}, \mathbf{g}_0 \in \mathbb{R}^n$ and $f_0 \in \mathbb{R}$ are independent of $\mathbf{x}$. In this case the equality constrained subproblem (6.1.7) becomes

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{Minimize} & f(\mathbf{x}^{(k)}) + \mathbf{d}^T\mathbf{g}^{(k)} + \frac{1}{2}\mathbf{d}^T G\mathbf{d} \\ Subject\ to & \mathbf{a}_i^T\mathbf{d} = 0 \quad i \in \mathcal{M}_+^{(k)}, \end{array} \tag{6.3.1}$$

where $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) = G\mathbf{x}^{(k)} + \mathbf{g}_0$ and $G = \nabla^2 f(\mathbf{x}^{(k)})$. The reduced gradient and reduced Hessian are

$$\mathbf{g}_Z^{(k)} = (Z^{(k)})^T \mathbf{g}^{(k)} \quad \text{and} \quad G_Z^{(k)} = (Z^{(k)})^T G Z^{(k)}$$

respectively. Note that $G_Z^{(k)}$ depends on $\mathbf{x}^{(k)}$ only through the dependence of $Z^{(k)}$ on $\mathbf{x}^{(k)}$. If the reduced Hessian is positive definite the solution to (6.3.1) is

$$\mathbf{d}^{(k)} = Z^{(k)} \mathbf{d}_Z^{(k)} \quad \text{where} \quad G_Z^{(k)} \mathbf{d}_Z^{(k)} = -\mathbf{g}_Z^{(k)}. \tag{6.3.2}$$

As G is symmetric the reduced Hessian is also symmetric. If $G_Z^{(k)}$ is positive definite then the system of linear equations $G_Z^{(k)} \mathbf{d}_Z^{(k)} = -\mathbf{g}_Z^{(k)}$ can be efficiently solved using a Cholesky factorization (see Section A.7.2 in Appendix A). If G is positive definite, so the problem is a *positive definite quadratic programming problem*, the reduced Hessian is always positive definite. A positive definite quadratic programming problem is a strictly convex programming problem, so any local minimizer is the global minimizer.

If $G_Z^{(k)}$ is indefinite then (6.3.1) does not have a finite minimum (except for the trivial case when $\mathbf{d} = 0$ is the only feasible point). This can only occur if G is indefinite, which is referred to as *indefinite quadratic programming*. An indefinite quadratic programming problem may have many different local minima. Finding the global minimum of an indefinite quadratic programming problem has been shown to be an NP-hard problem.

When G is positive definite the direction $\mathbf{d}^{(k)}$ given by (6.3.2) is the exact minimizer of (6.1.7). Thus a step of $\alpha = 1$, so $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$, should be taken if it produces a feasible point. To maintain feasibility the line search (6.1.8) is

$$\alpha^{(k)} = \min\left(1, \frac{-c_l^{(k)}}{\mathbf{a}_l^T \mathbf{d}^{(k)}}\right) = \min\left(1, \min_{i:\mathbf{a}_i^T \mathbf{d}^{(k)} > 0} \frac{-c_i(\mathbf{x}^{(k)})}{\mathbf{a}_i^T \mathbf{d}^{(k)}}\right). \tag{6.3.3}$$

If $\alpha^{(k)} = 1$, so the minimizer along the line is due to the curvature of the quadratic objective, *no* index is added to $\mathcal{M}^{(k)}$.

Example 6.3.1 *Solve the quadratic programming problem*

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^2}{\textit{Minimize}} & f(\mathbf{x}) = x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 + 2x_2 \\ \textit{Subject to} & c_1(\mathbf{x}) = x_1 + x_2 - 3 \le 0 \\ & c_2(\mathbf{x}) = -x_1 - 1 \le 0 \\ & c_3(\mathbf{x}) = x_1 - 2 \le 0 \\ & c_4(\mathbf{x}) = -x_2 \le 0 \end{array} \tag{6.3.4}$$

using an active set method starting from $\mathbf{x}^{(1)} = [\,0 \quad 2\,]^T$.

The feasible region, which is the same as in Example 6.2.1, and iterates are sketched in Figure 6.3.1. As all the constraints are inequalities $\mathcal{E} = \emptyset$ *and* $\mathcal{I} = \{1, \ldots, 4\}$. *The gradient and Hessian of the objective function are*

$$\mathbf{g}(x) = \begin{bmatrix} 2x_1 - 2x_2 - 4 \\ -2x_1 + 4x_2 + 2 \end{bmatrix}, \; G = \begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix},$$

while the constraint gradients are

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; \mathbf{a}_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \; \mathbf{a}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \; \mathbf{a}_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

The Hessian G *is positive definite as* $\det(G) = 4$ *and* $\operatorname{trace}(G) = 6$, *so this is a strictly convex quadratic programming problem.*

At $\mathbf{x}^{(1)}$ *the constraint values are* $\mathbf{c}^{(1)} = [\,-1 \quad -1 \quad -2 \quad -2\,]^T < 0^T$, *so* $\mathbf{x}^{(1)}$ *is in the interior of the feasible region, and the set of active constraints at* $\mathbf{x}^{(1)}$ *is* $\mathcal{A}^{(1)} = \emptyset$. *Hence* $\mathcal{M}^{(1)} = \emptyset$. *The objective value is* $f(\mathbf{x}^{(1)}) = 12$. *As* $\mathbf{g}^{(1)} = [\,-8 \quad 10\,]^T \neq 0$ *the search direction* $\mathbf{d}^{(1)}$ *is the solution of*

$$G\mathbf{d} = -\mathbf{g}^{(1)} \Longrightarrow \mathbf{d}^{(1)} = -G^{-1}\mathbf{g}^{(1)} = \frac{-1}{4}\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} -8 \\ 10 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}.$$

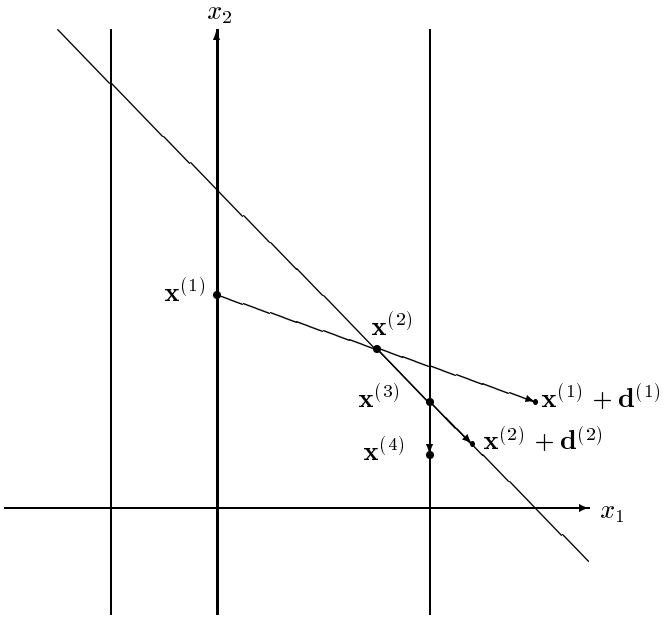


Figure 6.3.1: Quadratic programming example

A line search

i	1	2	3	4	*Curvature*
$c_i^{(1)}$	-1	-1	-2	-2	
$\mathbf{a}_i^T\mathbf{d}^{(1)}$	2	-3	3	1	
$\frac{-c_i^{(1)}}{\mathbf{a}_i^T\mathbf{d}^{(1)}} : \mathbf{a}_i^T\mathbf{d}^{(1)} > 0$	$\frac{1}{2}$		$\frac{2}{3}$	2	1

gives $\alpha^{(1)} = \frac{1}{2}$ *with* $j = 1$. *The second iterate is*

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha^{(1)}\mathbf{d}^{(1)} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ \frac{3}{2} \end{bmatrix},$$

where $f(\mathbf{x}^{(2)}) = -\frac{3}{4}$ *and* $\mathbf{g}^{(2)} = [\ -4 \quad 5\ ]^T$. *The new active set approximation is* $\mathcal{M}^{(2)} = \{1\}$. *The matrix of active constraint gradients is* $A^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. *Note that there are many possible choices for* $Z^{(2)} \neq 0$ *such that* $(Z^{(2)})^T A^{(2)} = 0$. *Using* $Z^{(2)} = [\ 1 \quad -1\ ]^T$

$$\mathbf{g}_Z^{(2)} = [\ 1 \quad -1\ ]\begin{bmatrix} -4 \\ 5 \end{bmatrix} = -9, \quad G_Z^{(2)} = [\ 1 \quad -1\ ]\begin{bmatrix} 2 & -2 \\ -2 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = 10.$$

Thus

$$G_Z^{(2)}\mathbf{d}_Z = \mathbf{g}_Z^{(2)} \Longrightarrow \mathbf{d}_Z^{(2)} = \frac{9}{10} \Longrightarrow \mathbf{d}^{(2)} = \frac{9}{10}\begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

A line search

i	1	2	3	4	*Curvature*
$c_i^{(2)}$	0	$\frac{-5}{2}$	$\frac{-1}{2}$	$\frac{-3}{2}$	
$\mathbf{a}_i^T\mathbf{d}^{(2)}$	0	$\frac{-9}{10}$	$\frac{9}{10}$	$\frac{9}{10}$	
$\frac{-c_i^{(2)}}{\mathbf{a}_i^T\mathbf{d}^{(2)}} : \mathbf{a}_i^T\mathbf{d}^{(2)} > 0$			$\frac{5}{9}$	$\frac{15}{9}$	1

gives $\alpha^{(2)} = \frac{5}{9}$ *with* $j = 3$. *The third iterate is*

$$\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha^{(2)}\mathbf{d}^{(2)} = \begin{bmatrix} \frac{3}{2} \\ \frac{3}{2} \end{bmatrix} + \frac{5}{9}\begin{bmatrix} \frac{9}{10} \\ \frac{-9}{10} \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

where $f(\mathbf{x}^{(3)}) = -4$ *and* $\mathbf{g}^{(3)} = [\,-2 \quad 2\,]^T$. *The new active set approximation is* $\mathcal{M}^{(3)} = \{1,3\}$. *As* $\mathbf{x}^{(3)}$ *is a vertex of the feasible region* $\mathbf{g}_Z^{(3)} = 0$. *The Lagrange multipliers are given by*

$$\sum_{i\in\mathcal{M}^{(3)}} \lambda_i \mathbf{a}_i + \mathbf{g}^{(3)} = 0 \Longrightarrow \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_3 \end{bmatrix} = -\begin{bmatrix} -2 \\ 2 \end{bmatrix} \Longrightarrow \begin{bmatrix} \lambda_1 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 4 \end{bmatrix}.$$

Hence

$$\lambda_j^{(3)} = \min_{i\in\mathcal{M}^{(3)}\cap\mathcal{I}} \lambda_i^{(3)} = -2 < 0,$$

with $j = 1$ *so* $\mathcal{M}_+^{(3)} = \{3\}$. *Using*

$$Z^{(3)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Longrightarrow \mathbf{g}_Z^{(3)} = 2,\ G_Z^{(3)} = 4 \Longrightarrow \mathbf{d}^{(3)} = \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix}.$$

A line search produces

i	1	2	3	4	*Curvature*
$c_i^{(3)}$	0	-3	0	-1	
$\mathbf{a}_i^T\mathbf{d}^{(3)}$	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	
$\frac{-c_i^{(3)}}{\mathbf{a}_i^T\mathbf{d}^{(3)}} : \mathbf{a}_i^T\mathbf{d}^{(3)} > 0$				2	1

so $\alpha^{(3)} = 1$ *with no new constraint becoming active. The fourth iterate is*

$$\mathbf{x}^{(4)} = \mathbf{x}^{(3)} + \alpha^{(3)}\mathbf{d}^{(3)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{1}{2} \end{bmatrix},$$

where $f(\mathbf{x}^{(4)}) = -\frac{9}{2}$ *and* $\mathbf{g}^{(4)} = [\,-1 \quad 0\,]^T$. *As the active set is* $\mathcal{M}^{(4)} = \mathcal{M}^{(3)} = \{3\}$ *one can use*

$$Z^{(4)} = Z^{(3)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Longrightarrow \mathbf{g}_Z^{(4)} = [\,0 \quad 1\,] \begin{bmatrix} -1 \\ 0 \end{bmatrix} = 0.$$

The multipliers are given by

$$\sum_{i\in\mathcal{M}^{(4)}} \lambda_i \mathbf{a}_i + \mathbf{g}^{(4)} = 0 \Longrightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \lambda_3 + \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Longrightarrow \lambda_3^{(4)} = 1 > 0.$$

Hence $\mathbf{x}^{(4)}$ *is the minimizer.*

6.3.1 Exercises

1. Consider the quadratic programming problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^2}{Minimiz} & f(\mathbf{x}) = 2x_1^2 - 4x_1x_2 + 6x_2^2 - 12x_1 - 4x_2 + 1 \\ Subject\ to & c_1(\mathbf{x}) = x_1 + x_2 - 5 \le 0 \\ & c_2(\mathbf{x}) = -x_1 + x_2 - 5 \le 0 \\ & c_3(\mathbf{x}) = -x_1 - 2 \le 0 \\ & c_4(\mathbf{x}) = x_1 - 4 \le 0 \\ & c_5(\mathbf{x}) = -x_2 - 1 \le 0 \end{array}$$

   (a) Show that the feasible region Ω is a convex set, and that f is a strictly convex quadratic function. Find the unconstrained global minimizer of f on $\mathbb{R}^2$.

   (b) Solve this problem using the active set method for quadratic programming starting from $\mathbf{x}^{(1)} = [\,1 \quad 1\,]^T$. Sketch the progress of the method.

2. Show that if the reduced Hessian $G_Z^{(k)}$ is positive definite and the reduced gradient $\mathbf{g}_Z^{(k)}$ is non-zero, then the search direction $\mathbf{d}^{(k)} = Z^{(k)}\mathbf{d}_Z^{(k)}$ obtained by solving $G_Z^{(k)}\mathbf{d}_Z^{(k)} = -\mathbf{g}_Z^{(k)}$ is a descent direction.

3. [**H**] Consider the problem

$$\begin{array}{c} Minimize \quad f(\mathbf{x}) \\ \mathbf{x} \in \Omega \end{array}$$

where $n = 6$,

$$\begin{aligned} f(\mathbf{x}) \quad = \quad & 448x_1x_2 + 828x_3x_4 + 448x_5x_6 - 144x_1x_4 - 64x_1x_6 - \\ & 144x_2x_3 - 144x_3x_6 - 64x_2x_5 - 144x_4x_5 - \\ & 120x_1 - 120x_2 - 270x_3 - 270x_4 - 120x_5 - 120x_6 + 735, \end{aligned}$$

and $\Omega \subset \mathbb{R}^n$ is defined by

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : 0 \leq x_i \leq 1, i = 1, \ldots, n\}.$$

(a) Calculate the gradient and Hessian of $f(\mathbf{x})$, and show that this is an indefinite quadratic programming problem.

(b) Show that $f(\mathbf{x})$ has an unconstrained stationary point $\mathbf{x}^*$ in the interior of the feasible region, and that $\mathbf{x}^*$ in nether a minimizer nor a maximizer of $f(\mathbf{x})$ on Ω.

(c) Show that

$$\overline{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

are strict local minimizers of $f(\mathbf{x})$ on Ω.

(d) It has been conjectured that $f(\mathbf{x})$ is non-negative on the feasible region. How could this be proved/disproved by solving an optimization problem.

(e) How many different active constraint sets need to be considered to find *all* the constrained stationary points.

(f) Write a MATLAB or `Maple` program to find all feasible constrained stationary points of $f(\mathbf{x})$ on Ω.

(g) Show that global extrema exist for this problem.

(h) Determine the global minimizer of $f(\mathbf{x})$ on Ω.

6.4 Interior Point Methods

Chapter 7

Methods for Nonlinearly Constrained Problems

This chapter considers the general nonlinear programming problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & f(\mathbf{x}) \\ Subject\ to & c_i(\mathbf{x}) = 0 \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}) \leq 0 \quad i \in \mathcal{I}, \end{array}$$

where some or all of the functions $c_i(\mathbf{x})$ may be nonlinear. Even the presence of a quadratic constraint can make the problem much more difficult than when there are only linear constraints. For linear constraints all iterates can be kept feasible once an initial feasible point has been found. However for nonlinear constraints a line search method will typically leave the feasible region as it attempts to follow a nonlinear constraint boundary (see Figure 7.0.1). Penalty or merit functions try to balance the reduction in the objective function with a reduction in the constraint violation. When moving around nonlinear constraint boundaries these may be incompatible, in that a substantial reduction in the objective function can only be achieved by allowing the point to become infeasible.

The two main approaches to nonlinearly constrained optimization are

- Use a penalty/merit function to convert the problem into an unconstrained problem which includes a parameter. The parameter is adjusted to force convergence of the unconstrained minimizer to a constrained stationary point. Given a value for the parameter the unconstrained merit function is minimised by an unconstrained method. The penalty parameter is then updated, and the unconstrained problem is minimized again, *starting from the last unconstrained minimizer.*

- The nonlinear constraints are linearized at the current iterate. A linear model of the objective then produces a linear programming subproblem to determine a search direction, while a quadratic model of the problem produces a quadratic programming problem. This is the basis of *sequential linear programming* (SLP) methods and *sequential quadratic programming* (SQP) methods respectively. The quadratic model should use or approximate the Hessian of the Lagrangian function, as constraint curvature can play an important role in determining a minimizer. These subproblems are used in conjunction with a line search or a trust region can be used to produce the next iterate.

7.1 Penalty and Merit Functions

Merit functions measure both the objective value and the constraint violation. For a single *equality* constraint $c_i(\mathbf{x}) = 0$ the constraint violation is $|c_i(\mathbf{x})|$. For a single *inequality* constraint $c_i(\mathbf{x}) \leq 0$ the constraint violation is the positive part $c_i^+(\mathbf{x}) = \max\{c_i(\mathbf{x}), 0\}$ of $c_i(\mathbf{x})$.

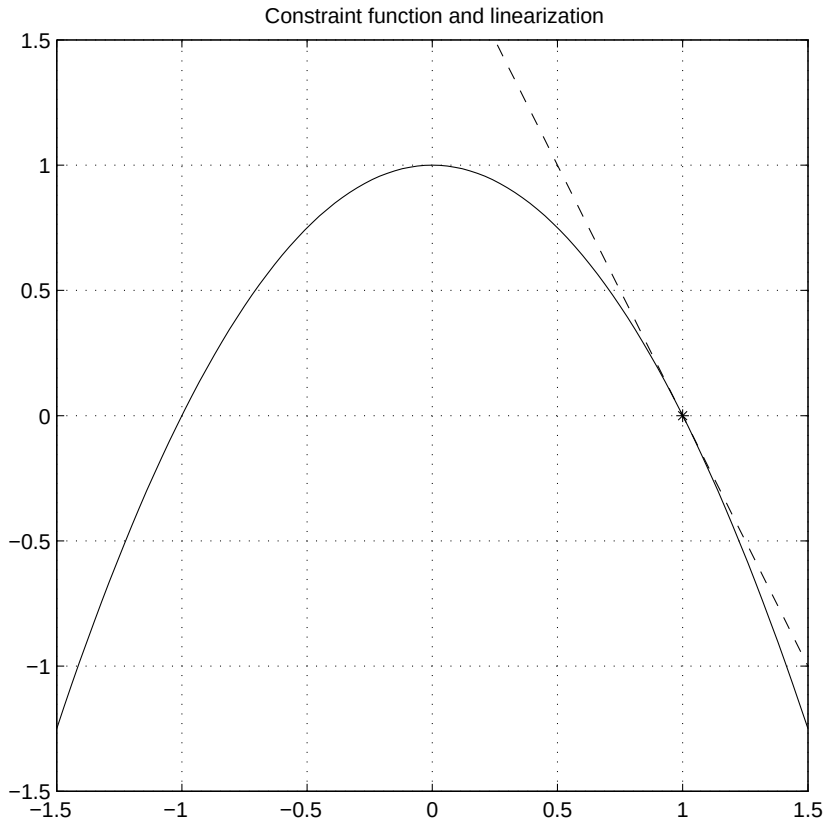


Figure 7.0.1: Linearization of $x_1^2 + x_2 - 1$ at $[1\ 0]^T$

Example 7.1.1 (Constraint violation) *Consider the constraint* $c_1(x) = x - 1 \leq 0$. *The constraint violation is*

$$c_1^+(x) = \max(c_1(x), 0) = \begin{cases} x - 1 & \text{if } x \geq 1, \\ 0 & \text{if } x \leq 1. \end{cases}$$

The functions $c_1(x)$, $c_1^+(x)$, $[c_1^+(x)]^2$ *and* $\nabla[c_1^+(x)]^2$ *are drawn in Figure 7.1.1.*

Assume that $\mathcal{E} = \{1, \ldots, m_E\}$ and $\mathcal{I} = \{m_E + 1, \ldots, m\}$. Let $\mathbf{c}_E(\mathbf{x}) = [\ c_1(\mathbf{x}) \quad \cdots \quad c_{m_E}(\mathbf{x})\ ]^T$ and Let $\mathbf{c}_I(\mathbf{x}) = [\ c_{m_E+1}(\mathbf{x}) \quad \cdots \quad c_m(\mathbf{x})\ ]^T$. A measure of how much a point $\mathbf{x}$ violates the constraints is

$$\left\| \begin{bmatrix} \mathbf{c}_E(\mathbf{x}) \\ \mathbf{c}_I^+(\mathbf{x}) \end{bmatrix} \right\|.$$

A penalty parameter σ is used to weight the constraint violations relative to the objective function. The use of a single penalty parameter σ assumes the constraints are equally scaled. If this is not the case the constraints may be rescaled, or separate penalty parameters σ_i for $i \in \mathcal{E} \cup \mathcal{I}$ used for each constraint. Any vector norm can be used, the most common being the 2-norm, 1-norm and ∞-norm.

- Sum of squares (ℓ_2) merit function:

$$\begin{aligned} \phi_2(\mathbf{x}; \sigma) &= f(\mathbf{x}) + \frac{\sigma}{2}\|\mathbf{c}_E(\mathbf{x})\|_2^2 + \frac{\sigma}{2}\|\mathbf{c}_I^+(\mathbf{x})\|_2^2 \\ &= f(\mathbf{x}) + \frac{\sigma}{2}\sum_{i \in \mathcal{E}}[c_i(\mathbf{x})]^2 + \frac{\sigma}{2}\sum_{i \in \mathcal{I}}[\max\{c_i(\mathbf{x}), 0\}]^2 \end{aligned}$$

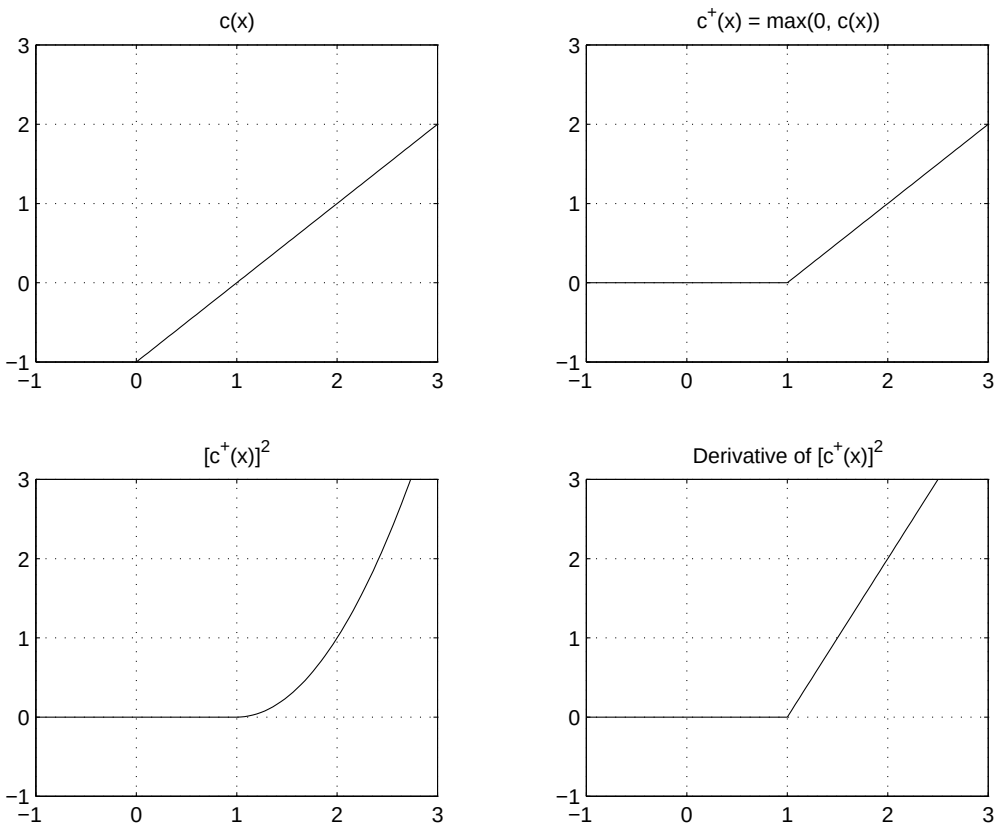


Figure 7.1.1: Inequality constraint violation

- ℓ_1 merit function:

$$\begin{aligned}\phi_1(\mathbf{x}.\sigma) &= f(\mathbf{x}) + \sigma\|\mathbf{c}_E(\mathbf{x})\|_1 + \sigma\|\mathbf{c}_I^+(\mathbf{x})\|_1 \\ &= f(\mathbf{x}) + \sigma\sum_{i\in\mathcal{E}}|c_i(\mathbf{x})| + \sigma\sum_{i\in\mathcal{I}}\max\{c_i(\mathbf{x}),0\}\end{aligned}$$

- ℓ_∞ merit function:

$$\begin{aligned}\phi_\infty(\mathbf{x};\sigma) &= f(\mathbf{x}) + \sigma\max\left\{\|\mathbf{c}_E(\mathbf{x})\|_\infty, \|\mathbf{c}_I^+(\mathbf{x})\|_\infty\right\} \\ &= f(\mathbf{x}) + \sigma\max\left\{|c_i(\mathbf{x})| : i\in\mathcal{E}, c_i(\mathbf{x}) : i\in\mathcal{I}, 0\right\}\end{aligned}$$

- Augmented Lagrangian merit function:

$$\begin{aligned}\phi_L(\mathbf{x};\lambda,\sigma) &= f(\mathbf{x}) + \lambda^T\mathbf{c}(\mathbf{x}) + \frac{\sigma}{2}\mathbf{c}_E(\mathbf{x})^T\mathbf{c}_E(\mathbf{x}) + \frac{\sigma}{2}\mathbf{c}_I^+(\mathbf{x})^T c_I^+(\mathbf{x}) \\ &= f(\mathbf{x}) + \sum_{i\in\mathcal{E}\cup\mathcal{I}}\lambda_i c_i(\mathbf{x}) + \frac{\sigma}{2}\sum_{i\in\mathcal{E}}[c_i(\mathbf{x})]^2 + \frac{\sigma}{2}\sum_{i\in\mathcal{I}}[\max\{c_i(\mathbf{x}),0\}]^2\end{aligned}$$

Both the ℓ_1 and ℓ_∞ merit functions are nonsmooth, that is not continuously differentiable at certain points (where $c_i(\mathbf{x}) = 0$ for some i, plus in the ℓ_∞ case where more than one constraint achieves the maximum). The sum of squares penalty function is twice continuously differentiable if there are no inequality constraints, but only once continuously differentiable if there are inequality constraints.

7.1.1 Sum of squares penalty function

Let $\mathbf{x}^*(\sigma^{(k)})$ solve

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \quad \phi_2(\mathbf{x}; \sigma^{(k)}).$$

Then it can be shown that $\mathbf{x}^*(\sigma^{(k)}) \to \mathbf{x}^*$ as $\sigma^{(k)} \to \infty$.

An unconstrained minimizer $\mathbf{x}^*(\sigma)$ of $\phi_2(\mathbf{x}; \sigma)$ must be a stationary point with $\nabla\phi(\mathbf{x}^*(\sigma); \sigma) = 0$. Thus

$$\nabla\phi_2(\mathbf{x}^*(\sigma); \sigma) = \nabla f(\mathbf{x}^*(\sigma)) + \sigma \sum_{i \in \mathcal{E}} c_i(\mathbf{x}^*(\sigma))\nabla c_i(\mathbf{x}^*(\sigma)) + \sigma \sum_{i \in \mathcal{I}} \max[c_i(\mathbf{x}^*(\sigma)), 0]\nabla c_i(\mathbf{x}^*(\sigma)) = 0.$$

At a feasible point $\mathbf{x}^*$ the indices of the active constraints are $\mathcal{A}(\mathbf{x}^*) = \mathcal{E} \cup (\mathcal{A}(\mathbf{x}^*) \cap \mathcal{I})$. At a constrained stationary point there exist multipliers λ_i^* for $i \in \mathcal{A}(\mathbf{x}^*)$ satisfying

$$\nabla f(\mathbf{x}^*) + \sum_{i \in \mathcal{E}} \lambda_i^* \nabla c_i(\mathbf{x}^*) + \sum_{i \in \mathcal{A}(\mathbf{x}^*) \cap \mathcal{I}} \lambda_i^* \nabla c_i(\mathbf{x}^*) = 0.$$

If $\mathbf{x}^*(\sigma) \to \mathbf{x}^*$ as $\sigma \to \infty$ then $\sigma c_i(\mathbf{x}^*(\sigma)) \to \lambda_i^*$. If $\mathbf{c}_i(\mathbf{x}^*) < 0$ then $c_1^+(\mathbf{x}^*) = 0$ and the corresponding multiplier is $\lambda_i^* = 0$. The Hessian of the sum of squares penalty function is

$$\begin{aligned} \nabla^2\phi(\mathbf{x}; \sigma) \quad = \quad & \nabla^2 f(\mathbf{x}) + \sigma \sum_{i \in \mathcal{E}} c_i(\mathbf{x})\nabla^2 c_i(\mathbf{x}) + \sigma \sum_{i \in \mathcal{I}} c_i^+(\mathbf{x})\nabla^2 c_i(\mathbf{x}) + \\ & \sigma \sum_{i \in \mathcal{E}} \nabla c_i(\mathbf{x})\nabla c_i(\mathbf{x})^T + \sigma \sum_{i \in \mathcal{I}} \begin{cases} \nabla c_i(\mathbf{x})\nabla c_i(\mathbf{x})^T & \text{if } c_i(\mathbf{x}) > 0 \\ 0 & \text{if } c_i(\mathbf{x}) < 0 \end{cases} \end{aligned}$$

As $\sigma \to \infty$ then $\sigma c_i(\mathbf{x}^*(\sigma)) \to \lambda_i$, so the first two terms give the Hessian of the Lagrangian function. However the other terms

$$\sigma \nabla c_i(\mathbf{x})\nabla c_i(\mathbf{x})^T$$

mean that the Hessian of the sum of squares penalty has elements which become very large. Thus the unconstrained minimization problems become numerically more and more difficulty to solve as σ becomes larger. Even the use of the previous value of $\mathbf{x}^*(\sigma^{(k-1)})$ as a starting point to minimize $\phi_2(\mathbf{x}; \sigma^{(k)})$ does not overcome this.

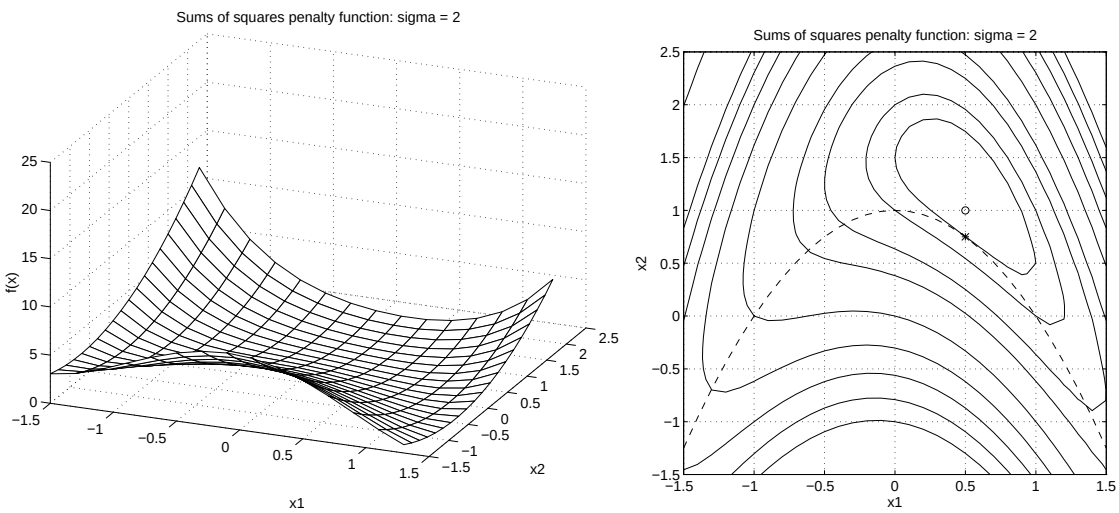


Figure 7.1.2: Surface and contour plots of $\phi_2(\mathbf{x}; 2)$

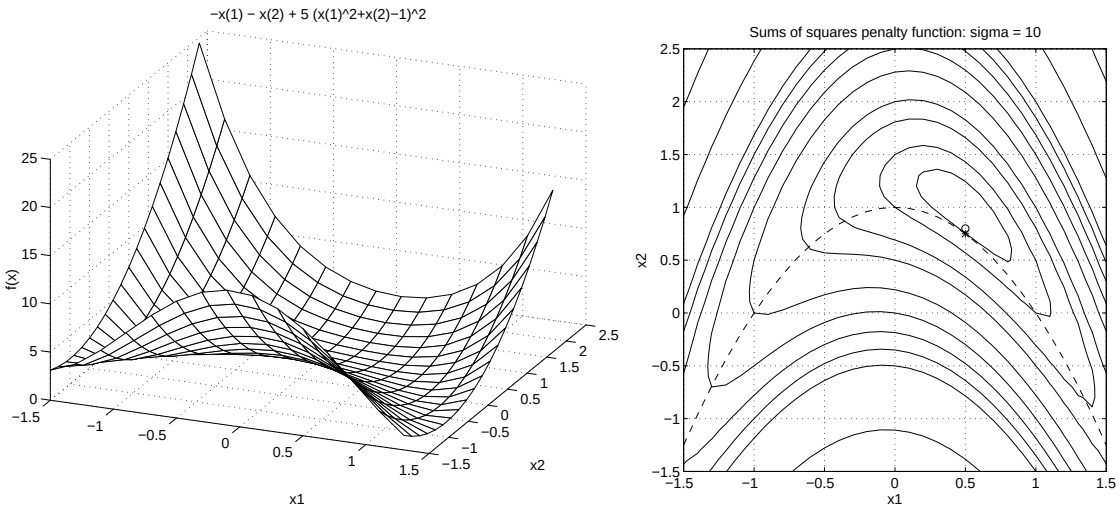


Figure 7.1.3: Surface and contour plots of $\phi_2(\mathbf{x}; 10)$

Example 7.1.2 (Sum of squares penalty function) *Consider the problem with $n = 2$, $\mathcal{E} = \emptyset$ $\mathcal{I} = \{1\}$ and and*

$$\begin{aligned} f(\mathbf{x}) &= -x_1 - x_2, \\ c_1(\mathbf{x}) &= x_1^2 + x_2 - 1. \end{aligned}$$

The sum of squares penalty functions $\phi(\mathbf{x}; 2)$ and $\phi_2(\mathbf{x}; 10)$ are drawn in Figures 7.1.2 and 7.1.3. The `Maple` *text file calculates the unconstrained minimizer of $\phi(\mathbf{x}; \sigma)$ and (drawn as a $\circ$ in the figures) and the limit as $\sigma \to \infty$ (the point $*$ in the figures).*

7.1.2 Augmented Lagrangian penalty function

The augmented Lagrangian penalty function produces convergence to a constrained stationary point by adjusting the Lagrange multiplier estimates so that $\lambda \to \lambda^*$, rather than taking $\sigma \to \infty$. This avoids the ill-conditioning of the Hessian of the penalty function. Let $\mathbf{x}^*(\lambda^{(k)}, \sigma)$ solve

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \quad \phi_L(\mathbf{x}; \lambda^{(k)}, \sigma).$$

As $\mathbf{x}^*(\lambda^{(k)}, \sigma)$ is a stationary point of $\phi_L(\mathbf{x}; \lambda, \sigma)$

$$\begin{aligned} \nabla_{\mathbf{x}} \phi_L(\mathbf{x}^*(\lambda^{(k)}, \sigma); \lambda^{(k)}, \sigma) &= \nabla f(\mathbf{x}^*(\lambda^{(k)}, \sigma)) + \sum_{i \in \mathcal{E}} \left(\lambda_i^{(k)} + \sigma c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)) \right) \nabla c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)) + \\ & \sum_{i \in \mathcal{I}} \left(\lambda_i^{(k)} + \sigma \max\{c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)), 0\} \right) \nabla c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)). \end{aligned}$$

Comparing with the conditions for a constrained stationary points show that

$$\begin{aligned} \lambda_i^{(k+1)} &= \lambda^{(k)} + \sigma c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)) \quad i \in \mathcal{E}, \\ \lambda_i^{(k+1)} &= \lambda^{(k)} + \sigma \max\{c_i(\mathbf{x}^*(\lambda^{(k)}, \sigma)), 0\} \quad i \in \mathcal{I}, \end{aligned}$$

provide new estimates of the Lagrange multipliers λ^*. Then for $\sigma > \bar{\sigma}$, $\mathbf{x}^*(\lambda^{(k)}, \sigma) \to \mathbf{x}^*$ as $\lambda^{(k)} \to \lambda^*$.

7.1.3 Exact penalty functions

The penalty functions $\phi_1(\mathbf{x};\sigma)$ and $\phi_\infty(\mathbf{x};\sigma)$ are referred to as *exact penalty functions* as it can be show that for σ sufficiently large a local minimizer of the penalty function must satisfy the first order necessary conditions for a local minimizer of the constrained problem. Thus there is no need to take a sequence of penalty parameters (or sequence of Lagrane multiplier estimates) to get convergence to a constrained stationary point. The problems with exact penalty functions are

- The functions $\phi_1(\mathbf{x};\sigma)$ and $\phi_\infty(\mathbf{x};\sigma)$ are nonsmooth. Moreover the points where they are nondifferentiable include the constraint boundaries, which is typically where the constrained minimizer lies. Thus methods for minimizing these functions must be able to handle derivative discontinuities. Such methods are discussed in Chapter 10.
- Just how large a value of σ is needed. For the ℓ_1 penalty function it can be shown that

$$\sigma \geq \bar{\sigma} \equiv \|\lambda^*\|_\infty$$

is sufficient. For the ℓ_∞ penalty function it can be shown that

$$\sigma \geq \bar{\sigma} \equiv \|\lambda^*\|_1$$

is sufficient. Unfortunately the lagrange multipliers λ^* at the solution are not known before the problem is solved. Lagrange multiplier estimates can be used as the basis for a scheme for increasing $\sigma^{(k)}$.

7.1.4 Exercises

1. Consider the problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{Minimize} & f(\mathbf{x}) \\ Subject\ to & c_i(\mathbf{x}) = 0 \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}) \leq 0 \quad i \in \mathcal{I}, \end{array}$$

where $n = 2$, $\mathcal{E} = \emptyset$, $\mathcal{I} = \{1, 2\}$, and

$$\begin{aligned} f(\mathbf{x}) &= -x_1 + x_2, \\ c_1(\mathbf{x}) &= x_1^2 + x_2 - 1, \\ c_2(\mathbf{x}) &= -x_2 - 1. \end{aligned}$$

Let $\mathbf{x}^{(1)} = 0$.

(a) Show that $\mathbf{x}^* = [\ \sqrt{2} \quad -1\ ]^T$ is the global minimizer. What are the values of λ^* and $f(\mathbf{x}^*)$.

(b) FInd the minimizer of the sum of squares penalty function for $\sigma = 1, 10, 100, 1000$ and compare the values of $\mathbf{x}^*(\sigma)$, $\phi(\mathbf{x}^*(\sigma))$, $\nabla^2\phi(\mathbf{x}^*(\sigma))$, $\mathbf{c}(\mathbf{x}^*(\sigma))$ and $\sigma\mathbf{c}(\mathbf{x}^*(\sigma))$ with their theoretical limits as $\sigma \to \infty$.

(c) Use an augmented Lagrangian penalty function with $\sigma = 10$ and four iterates generate by

$$\lambda^{(k+1)} = \lambda^{(k)} + \sigma\mathbf{c}(\mathbf{x}^*(\lambda^{(k)}, \sigma))$$

starting from $\lambda^{(1)} = 0$.

7.2 Sequential quadratic programming

7.2.1 Sequential linear programming

Linearizing the constraints about the current iterate gives

$$\begin{aligned} c_i(\mathbf{x}) = 0 \quad &\Longrightarrow \quad c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) = 0 \quad i \in \mathcal{E}, \\ c_i(\mathbf{x}) \leq 0 \quad &\Longrightarrow \quad c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) \leq 0 \quad i \in \mathcal{I}. \end{aligned}$$

If the objective function is also linearized this produces the linear programming subproblem

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{Minimize} & f^{(k)} + \mathbf{d}^T\mathbf{g}^{(k)} \\ Subject\ to & c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) = 0 \quad i \in \mathcal{E} \\ & c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) \le 0 \quad i \in \mathcal{I}. \end{array} \tag{7.2.1}$$

The subproblem may not have a finite solution (see Example 7.2.1) so a trust region

$$\|\mathbf{d}\| \le \delta$$

must be added. If the 2-norm is used the trust region constraint represents a quadratic constraint, which destroys the linear programming structure of (7.2.1). Using the ∞-norm corresponds to the constraints

$$\|\mathbf{d}\|_\infty \le \delta \iff -\delta \le d_i \le \delta$$

which are simple lower and upper bounds on the variables. These constraints can be added to the linear programming subproblem (7.2.1) keeping the linear programming structure. In fact most good linear programming algorithms can exploit the special structure of the simple lower and upper bounds.

The search direction $\mathbf{d}^{(k)}$ obtained by solving (7.2.1) can be used in a line search

$$\alpha^{(k)} = argmin\ \phi(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}; \sigma)$$

using a penalty or merit function. Sequential linear programming methods do not include any curvature information from the objective function or the constraints. Thus they only converge superlinearly if the solution is a vertex of the feasible region.

Example 7.2.1 (SLP method for a nonlinear constraint)

Consider the problem with $n = 2$, $\mathcal{E} = 1$, $\mathcal{I} = \emptyset$, and

$$\begin{aligned} f(\mathbf{x}) &= -x_1 - x_2 \\ c_1(\mathbf{x}) &= x_1^2 + x_2 - 1. \end{aligned}$$

This problem has

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \nabla c_1(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 1 \end{bmatrix}.$$

Linearizing the constraint at $\mathbf{x}^{(1)} = [\ 1 \quad 0\ ]^T$ gives $c_1(\mathbf{x}^{(1)}) = 0$ and $\nabla c_1(\mathbf{x}^{(1)}) = [\ 2 \quad 1\ ]^T$, so

$$c_1(\mathbf{x}^{(1)}) + \mathbf{d}^T\nabla c_1(\mathbf{x}^{(1)}) = 0 + 2d_1 + d_2 = 0.$$

The linear programming subproblem (7.2.1) is

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^2}{Minimize} & -1 - d_1 - d_2 \\ Subject\ to & 2d_1 + d_2 = 0. \end{array} \tag{7.2.2}$$

The subproblem (7.2.2) does not have a finite solution as $\mathbf{d} = [\ d_1 \quad -2d_1\ ]^T$ is feasible for any d_1, and the objective function is $-1 + d_1 \to -\infty$ as $d_1 \to -\infty$.

Adding the trust region

$$\|\mathbf{d}\|_\infty \le 1$$

produces the subproblem

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^2}{Minimize} & -1 - d_1 - d_2 \\ Subject\ to & 2d_1 + d_2 = 0 \\ & -1 \le d_1 \le 1 \\ & -1 \le d_2 \le 1. \end{array} \tag{7.2.3}$$

which has the solution $\mathbf{d}^{(k)} = [\ -\frac{1}{2} \quad 1\ ]^T$ with subproblem objective value $-\frac{3}{2}$. The new point

$$\mathbf{x}^{(k)} + \mathbf{d}^{(k)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

has $c_1(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}) = \frac{1}{4}$ so is not feasible.

7.2.2 Sequential quadratic programming

A common *regularization* technique is to add the term

$$\eta\|\mathbf{x}-\mathbf{x}^{(k)}\|_2^2$$

to the objective of the subproblem used to determine the search direction. This produces the subproblem

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{Minimize} & f^{(k)}+\mathbf{d}^T\mathbf{g}^{(k)}+\frac{\eta}{2}\mathbf{d}^T\mathbf{d} \\ Subject\ to & c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})=0 \quad i\in\mathcal{E} \\ & c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})\le 0 \quad i\in\mathcal{I}. \end{array} \tag{7.2.4}$$

The Hessian of the quadratic model of the objective function is

$$H^{(k)}=\eta I.$$

For $\eta>0$ the subproblem (7.2.4) is a strictly convex quadratic programming problem, so a global minimizer can be readily found.

The subproblem (7.2.4) includes a curvature term, but not the curvature of the objective or constraints. From the optimality conditions the appropriate way to include the curvature of the objective and the constraints is through the Hessian

$$\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x};\lambda)=\nabla^2 f(\mathbf{x})+\sum_{i\in\mathcal{E}\cup\mathcal{I}}\lambda_i\nabla^2 c_i(\mathbf{x})$$

of the Lagrangian function. Using

$$H^{(k)}=\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{(k)},\lambda^{(k)})$$

gives the quadratic programming subproblem

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{Minimize} & q_k(\mathbf{d})=f^{(k)}+\mathbf{d}^T\mathbf{g}^{(k)}+\frac{1}{2}\mathbf{d}^TH^{(k)}\mathbf{d} \\ Subject\ to & c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})=0 \quad i\in\mathcal{E} \\ & c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})\le 0 \quad i\in\mathcal{I}. \end{array} \tag{7.2.5}$$

The necessary conditions for a local minimizer guarantee that the reduced Hessian $(Z^*)^T\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*;\lambda^*)Z^*$ is positive semi-definite. However the full Hessian of the Lagrangian may be indefinite. Thus (7.2.5) may be an indefinite quadratic programming problem, with many local minimizers, and finding a global minimizer may be very difficult.

A solution $\mathbf{d}^{(k)}$ to the subproblem (7.2.5) must satisfy

$$\begin{aligned} &\mathbf{g}^{(k)}+H^{(k)}\mathbf{d}^{(k)}+\sum_{i\in\mathcal{A}(\mathbf{x}^{(k)})}\lambda_i^{(k)}\nabla c_i(\mathbf{x}^{(k)})=0, \\ &c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})=0 \quad i\in\mathcal{E}, \\ &c_i(\mathbf{x}^{(k)})+\mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)})\le 0 \quad i\in\mathcal{I}, \\ &\lambda_i^{(k)}\ge 0 \quad i\in\mathcal{A}(\mathbf{x}^{(k)})\cap\mathcal{I}. \end{aligned}$$

If $\mathbf{d}^{(k)}=0$ is a solution then

$$\begin{aligned} &\mathbf{g}^{(k)}+\sum_{i\in\mathcal{A}(\mathbf{x}^{(k)})}\lambda_i^{(k)}\nabla c_i(\mathbf{x}^{(k)})=0, \\ &c_i(\mathbf{x}^{(k)})=0 \quad i\in\mathcal{E}, \\ &c_i(\mathbf{x}^{(k)})\le 0 \quad i\in\mathcal{I}, \\ &\lambda_i^{(k)}\ge 0 \quad i\in\mathcal{A}(\mathbf{x}^{(k)})\cap\mathcal{I}. \end{aligned}$$

so $\mathbf{x}^{(k)}$ satisfies the first order necessary conditions for a local minimizer. Thus a common convergence test for sequential quadratic programming methods is $\|\mathbf{d}^{(k)}\| \leq \epsilon_{\mathbf{d}}$.

Another issue is what values should be used for the Lagrange multiplier estimate $\lambda^{(k)}$ at the current iterate $\mathbf{x}^{(k)}$. The Lagrange multipliers $\lambda^{(k)}$ obtained from the solution of the linear or quadratic programming subproblem provide estimates of the the Lagrange multipliers λ^*. However $\lambda^{(k)}$ is only available after the subproblem has been solved. Thus the multiplier estimates used in calculating the Hessian of the Lagrangian must either come from the solution of a previous subproblem, so

$$H^{(k)} = \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(k)}, \lambda^{(k-1)}),$$

or $\lambda^{(k)}$ must be estimated by calculating the least squares solution of the (typically over-determined) system of linear equations

$$\mathbf{g}^{(k)} + \sum_{i \in \mathcal{A}(\mathbf{x}^{(k)})} \lambda_i^{(k)} \nabla c_i(\mathbf{x}^{(k)}) = 0.$$

If an estimate $\mathcal{A}(\mathbf{x}^{(k)})$ of the active constraints is not known the the linearly constrained least squares problem

$$\begin{array}{ll} \underset{\lambda}{Minimize} & \|\mathbf{g}^{(k)} + \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i \nabla c_i(\mathbf{x}^{(k)})\|^2 \\ Subject\ to & \lambda_i \geq 0 \quad i \in \mathcal{I}. \end{array} \tag{7.2.6}$$

can be solved to provide multiplier estimates.

The search direction subproblem may then be used in either a line search algorithm or a trust region algorithm. In both cases a *merit* function is used to gauge progress, not just the objective function.

The disadvantages of using the full Hessian of the Lagrangian function are

- the objective Hessian $\nabla^2 f(\mathbf{x}^{(k)})$ and constraint Hessians $\nabla^2 c_i(\mathbf{x}^{(k)})$ must be evaluated;
- the Hessian of the Lagrangian may be indefinite, so the quadratic programming subproblem is not a convex programming problem.

One strategy to try to overcome these problems is to use a symmetric positive-definite approximation $B^{(k)}$ to the Hessian of the Lagrangian function. The approximation is updated using quasi-Newton updates with

$$\begin{aligned} \mathbf{s}^{(k)} &= \alpha^{(k)} \mathbf{d}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} &= \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(k+1)}; \lambda^{(k+1)}) - \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(k)}; \lambda^{(k)}). \end{aligned}$$

Using differences in the gradients of the Lagrangian means that $B^{(k)}$ will approximate the Hessian of the Lagrangian. The quasi-newton updating formulae guarantee that $B^{(k+1)}$ is positive definite if $B^{(k)}$ is positive definite and ${\mathbf{s}^{(k)}}^T \mathbf{y}^{(k)} > 0$. Here a line search on a merit function (for example the augmented Lagrangian penalty function) does not guarantee that ${\mathbf{s}^{(k)}}^T \mathbf{y}^{(k)} > 0$. Various strategies have been suggested (see [Fle87]), including taking $B^{(k+1)} = B^{(k)}$ if ${\mathbf{s}^{(k)}}^T \mathbf{y}^{(k)} \leq 0$.

Again the issue of which Lagrange multiplier estimated should be used in the calculation of $\mathbf{y}^{(k)}$ arises. Either multiplier estimates from previous quadratic programming subproblems must be used, so

$$\mathbf{y}^{(k)} = \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(k+1)}; \lambda^{(k)}) - \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^{(k)}; \lambda^{(k-1)}),$$

or a linear least squares problem must be solved to provide the multiplier estimates $\lambda^{(k+1)}$.

If $B^{(k)}$ is positive definite then the subproblem is a strictly convex quadratic programming problem for which a global minimizer can be readily found. The direction $\mathbf{d}^{(k)}$ from the quadratic programming subproblem is typically used in a line search based on a merit function, for example the augmented Lagrangian penalty function so

$$\alpha^{(k)} = argmin\ \phi_L(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}; \sigma).$$

Alternatively a trust region method can be implemented. As in sequential linear programming using the infinity norm trust region $\|\mathbf{d}\|_\infty \le \delta$ gives the quadratic programming subproblem

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{Minimize} & q_k(\mathbf{d}) = f^{(k)} + \mathbf{d}^T\mathbf{g}^{(k)} + \frac{1}{2}\mathbf{d}^T B^{(k)}\mathbf{d} \\ Subject\ to & c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) = 0 \quad i\in\mathcal{E} \\ & c_i(\mathbf{x}^{(k)}) + \mathbf{d}^T\nabla c_i(\mathbf{x}^{(k)}) \le 0 \quad i\in\mathcal{I} \\ & \\ & -\delta \le d_i \le \delta \quad i = 1,\ldots,n. \end{array} \tag{7.2.7}$$

The trust region radius δ should be updated based on the agreement between the subproblem model and a merit function.

7.2.3 Exercises

1. Consider the problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{Minimize} & f(\mathbf{x}) \\ Subject\ to & c_i(\mathbf{x}) = 0 \quad i\in\mathcal{E} \\ & c_i(\mathbf{x}) \le 0 \quad i\in\mathcal{I}, \end{array}$$

where $n = 2$, $\mathcal{E} = \emptyset$, $\mathcal{I} = \{1, 2\}$, and

$$\begin{aligned} f(\mathbf{x}) &= -x_1 + x_2, \\ c_1(\mathbf{x}) &= x_1^2 + x_2 - 1, \\ c_2(\mathbf{x}) &= -x_2 - 1. \end{aligned}$$

Let $\mathbf{x}^{(1)} = 0$. This is the same problem as in Exercise 1 on page 170.

(a) Solve this problem starting from $\mathbf{x}^{(1)}$ using a sequential linear programming method using a trust region $\|\mathbf{d}\|_\infty \le 1$.

(b) Show that

i. $\mathbf{x}^*$ is a vertex of the feasible region satisfying strict complementarity.

ii. the sequential linear programming method converges quadratically to $\mathbf{x}^*$.

Chapter 8

Portfolio Optimization [H]

8.1 Introduction

Portfolio optimization problems or asset allocation problems look at the "best" way for an investor or fund manager to allocate funds between a number of different asset classes. This area started with the work of Markowitz [Mar52, Mar59, Mar87], and active interest has continued since then (see [Hau97, KLM84, LH89, Mer71, Rei94, Rud88, Sha70b, Zen93, BKM96, Lue97] for example).

Financial assets are typically divided into a number of classes, for example

- Cash and equivalents.
- Corporate bonds.
- Equities.

The return on an asset is calculated by

$$\text{Return} = \frac{\text{Final price of asset} - \text{Initial price of asset}}{\text{Initial price of asset}},$$

and is usually expressed as a percentage by multiplying the above ratio by 100. Each asset class typically offers a different return and also entails a different risk. For example a Cash Management Trust may offer a return of 7% with very low risk, while shares may offer the possibility of returns of 20% but with a high risk of obtaining a much lower return.

The value of a financial instrument is modelled as a random variable - that is it involves an unknown component which is stochastic. The value of each asset class is characterized by the distribution of the random variable.

- The *return* is the expected value (mean) of the distribution.
- A measure of *risk* is the variance of the distribution.

A risk free investment is one with a zero variance, while a risky investment is one with a large variance. The distribution of a nearly risk free investment and a risky investment, both with an expected return of 12%, are illustrated in Figure 8.1.1. Other measures of risk are discussed in Section 8.4.

An outline of a strategy used to manage a portfolio of investments is

1. Select categories of financial instruments.
2. Allocate portion of capital to each category.
3. Monitor and adjust the allocation if necessary.

The basic assumptions are that the players in the market wish to

1. *Maximize their return from investments.*

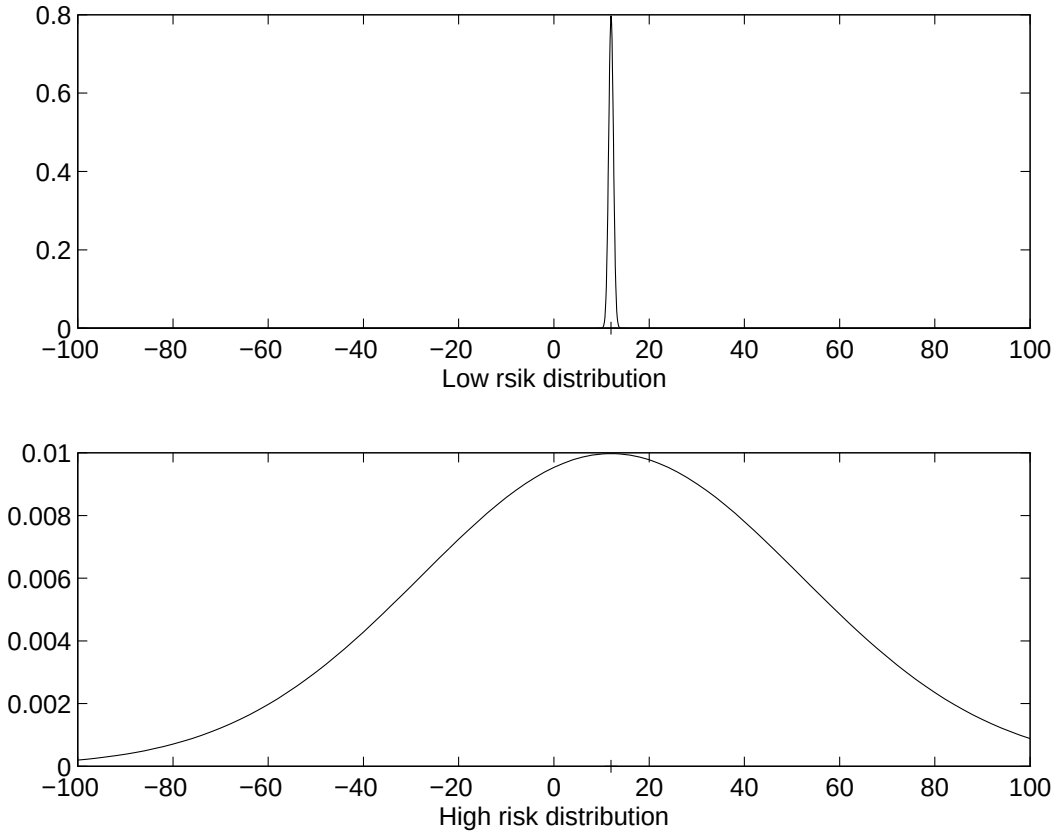


Figure 8.1.1: Low and high risk investments

2. *Avoid risk: Given a choice between two assets with equal returns the asset with the lowest risk is selected.*

Consider Figure 8.1.2 where the return and risk of four assets A_1, A_2, A_3 and A_4 are plotted. From Assumption 1 asset A_1 is preferable to A_2 as it gives a greater return for the same risk. Similarly A_3 is preferable to A_4. From Assumption 2 asset A_2 is preferable to A_4 as it gives the same return but at a lower risk. However it is not clear from these assumption whether A_1 is preferable to A_3; that depends on individual preferences (are you willing to run a higher risk of loosing all your capital for the chance of a higher return?).

There are several ways of handling many competing objective functions.

1. Find a suitable weighting of the objective to combine them into a single objective.

   For example a risk averse (conservative) investor may maximize

   $$Objective = 0.2 \times Return - 0.8 \times Risk$$

   while an investor willing to run more risk seeking a larger return may use

   $$Objective = 0.9 \times Return - 0.1 \times Risk$$

   Both these objectives assume $Return$ and $Risk$ are measured in comparable units.

2. Use some of the objectives to impose constraints on the problem, limiting the values those objective functions can take.

   For example an investor could maximize $Return$ subject to the constraint that $Risk \leq \nu$. A Risk averse investor would chose a small value of ν, while an investor willing to take more risk would chose a large value for ν.

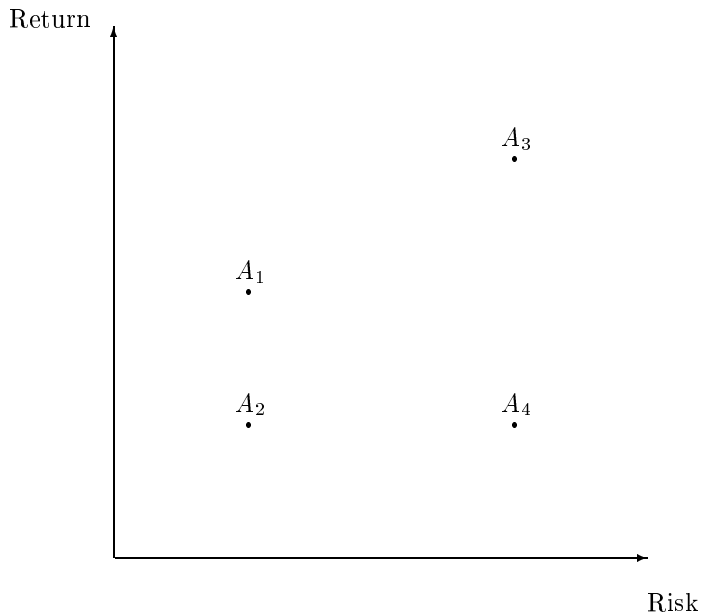


Figure 8.1.2: Return vs risk of four assets

A common approach is to calculate the *efficient frontier*, which is the curve giving the minimum risk portfolio for each specified return. An example is give in Figure 8.1.3 The section of the curve in which the risk increases with increasing return is call the efficient frontier, while the section in which the risk increases with decreasing return is called the inefficient frontier. The efficient frontier can be calculated by solving a parametric optimization discussed in Section 8.2.2.

One of the key ideas coming from the pioneering work of Markowitz in 1952 is that diversifying a portfolio among several (many) classes is a means of controlling risk. To establish this, and to formulate optimization problems that attempt to allocate a portfolio amongst several assets classes, the basic statistical concepts in Appendix B should be reviewed.

8.1.1 Asset Portfolios

Let $A_1, \ldots, A_n$ be assets classes. The aim is to allocate a portion of a portfolio to each asset class. For $i = 1, \ldots, n$ let

$$x_i = \text{fraction of portfolio allocated to asset class } A_i.$$

This definition of the variables implies that

$$x_i \geq 0 \quad i = 1, \ldots, n \qquad \text{and} \qquad \sum_{i=1}^{n} x_i = 1. \tag{8.1.1}$$

Care must be taken with units as the portion of the portfolio allocated to an asset is typically quoted as a percentage. A portfolio management problem is to determine the "best" set of x_i subject to all the required conditions.

A manager may also have both lower and upper bounds on the percentage of the portfolio they are willing to allocate to a given asset class. This corresponds to simple lower and upper bounds

$$\ell_i \leq x_i \leq u_i \quad i = 1, \ldots, n \tag{8.1.2}$$

on the variables. For example a constraint that the fraction of a portfolio allocated to property investments must be between 5% and 20% corresponds to

$$0.05 \leq x_i \leq 0.20,$$

where the variable x_i represents the fraction of the portfolio allocated to property.

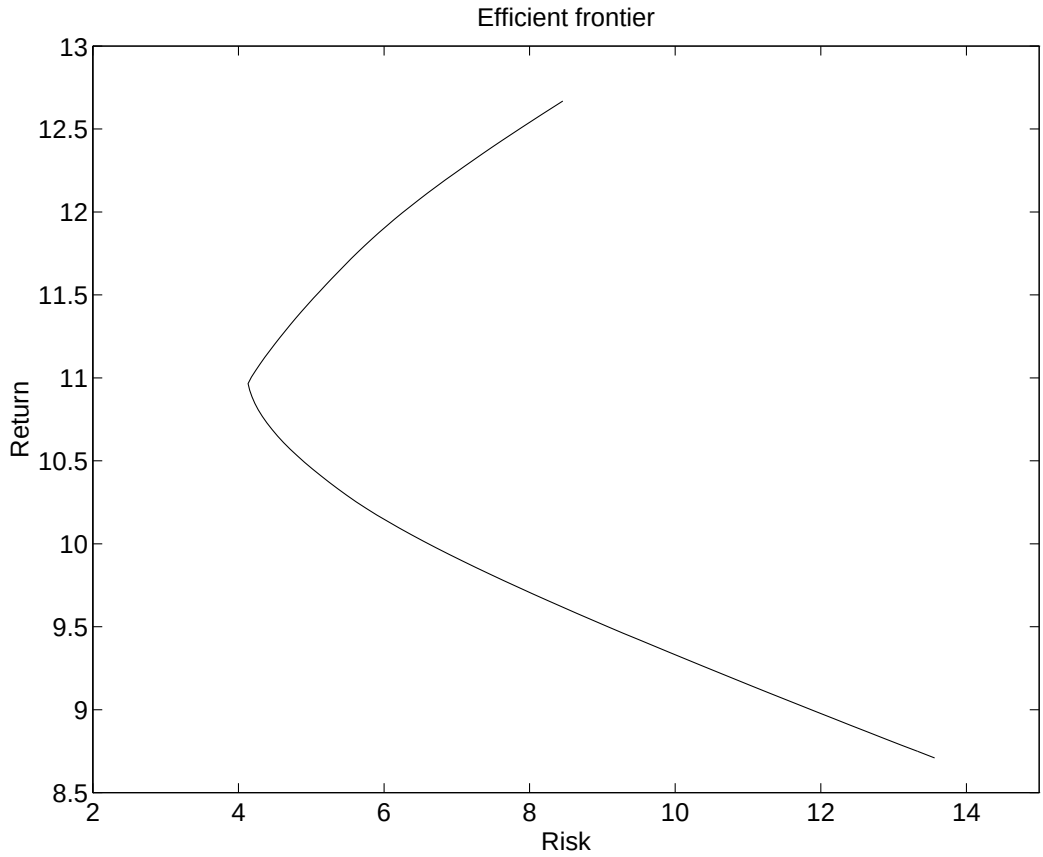


Figure 8.1.3: Efficient and inefficient frontiers

There may also be more general linear constraints which can be represented as

$$\ell_{n+i} \leq \mathbf{a}_i^T \mathbf{x} \leq u_{n+i} \quad i = 1, \ldots, m. \tag{8.1.3}$$

For example if x_1, x_2, x_3 correspond to different international asset classes, then a constraint that the fraction of the portfolio allocated to all international assets must not exceed 40% gives

$$x_1 + x_2 + x_3 \leq 0.40.$$

The simple lower and upper bounds (8.1.2) and general linear constraints (8.1.3) can be written as

$$\ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u, \tag{8.1.4}$$

where $\ell \leq u \in \mathbb{R}^{n+m}$. Equality constraints can be represented by setting $\ell_i = u_i$. The linear constraints (8.1.4) define a convex feasible region in $\mathbb{R}^n$.

Let the return from each asset class be a random variable X_i with a mean μ_i and standard deviation σ_i. This does not imply that these random variables are normally distributed. The portfolio is modelled by a random variable P given by

$$P = \sum_{i=1}^{n} x_i X_i.$$

The expected return on the portfolio P is the

$$\mathcal{E}[\,P\,] = \mathcal{E}[\sum_{i=1}^{n} x_i X_i\,] = \sum_{i=1}^{n} x_i \mathcal{E}[\,X_i\,] = \sum_{i=1}^{n} x_i \mu_i. \tag{8.1.5}$$

The variance of the portfolio P is

$$\mathcal{V}[P] = \mathcal{V}[\sum_{i=1}^{n} x_i X_i] = \sum_{i=1}^{n} x_i^2 V(X_i) + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_i x_j Cov[X_i, X_j]. \tag{8.1.6}$$

Define the *covariance matrix* $C \in \mathbb{R}^{n\times n}$ by

$$C_{ij} = Cov[X_i, X_j] \qquad i = 1, \ldots, n \quad j = 1, \ldots, n.$$

The definition of covariance immediately implies C is a symmetric matrix. Moreover the diagonal elements are

$$C_{ii} = Cov[X_i, X_i] = \mathcal{V}[X_i] = \sigma_i^2 \qquad i = 1, \ldots, n.$$

The *correlation matrix* K is the n by n symmetric matrix defined by

$$K_{ij} = Cor[X_i, X_j] = \frac{Cov[X_i, X_j]}{\sigma_i \sigma_j} \qquad i = 1, \ldots, n \quad j = 1, \ldots, n.$$

Note that

$$K_{ii} = 1 \qquad i = 1, \ldots, n.$$

Let $\Sigma = \text{diag}(\,)\sigma_1, \ldots, \sigma_n)$. Then

$$C = \Sigma K \Sigma \tag{8.1.7}$$

as pre-multiplying K by the diagonal matrix Σ scales the ith row of the matrix K by σ_i, and post-multiplying K by the diagonal matrix Σ scales the jth column of K by σ_j.

Let $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ be the vector of variables. Then (8.1.6) is

$$\mathcal{V}[P] = \mathbf{x}^T C \mathbf{x} \tag{8.1.8}$$

As the variance is nonnegative this implies that the covariance matrix C must be positive semi-definite.

8.2 Mean-variance models

This section looks at several models which are based on using the variance of a portfolio as a measure of the risk of the portfolio. Other measures of risk are considered in section 8.4.

8.2.1 Maximizing Returns

Let $\mathbf{r} = [\ \mu_1 \quad \ldots \quad \mu_n\ ]^T \in \mathbb{R}^n$. To maximize the return from the portfolio we want to maximize $\mathcal{E}[P] = \mathbf{r}^T\mathbf{x}$. Let $\mathbf{e} = [\ 1 \quad \ldots \quad 1\ ]^T \in \mathbb{R}^n$. The constraints (8.1.1) on the variables can be written as $0 \le \mathbf{x} \le \mathbf{e}$ and $\mathbf{e}^T\mathbf{x} = 1$, where the constraints $x_i \le 1$ are an immediate consequence of (8.1.1). This produces the optimization problem

$$\begin{array}{ll} \textit{Maximize} & \mathbf{r}^T\mathbf{x} \\ \mathbf{x} \in \mathbb{R}^n & \\ \textit{Subject to} & \mathbf{e}^T\mathbf{x} = 1 \\ & 0 \le \mathbf{x} \le \mathbf{e}. \end{array} \tag{8.2.1}$$

Example 8.2.1 *Let the expected returns from* 14 *asset classes be given by the vector* $\mathbf{r} \in \mathbb{R}^{14}$, *where*

$$\mathbf{r} = [\ 5 \quad 2 \quad 4 \quad 6 \quad 8 \quad 12 \quad 1 \quad 3 \quad 5 \quad 6 \quad 12 \quad 4 \quad 12 \quad 5\ ]^T. \tag{8.2.2}$$

1. *What is a solution to (8.2.1) when* $\mathbf{r}$ *is defined by (8.2.2)?*
2. *What is the set of all solutions to (8.2.1) when* $\mathbf{r}$ *is defined by (8.2.2)?*
3. *What is the general solution to (8.2.1)?*

Answer

1. *The variables* $x_i \in [0,1]$ *represent the fraction of the portfolio that is allocated to asset class* A_i. *Thus a solution is obtained by putting* all *the portfolio into the asset class with the largest expected return, which is* 12 *for the expected returns in (8.2.2). Thus a solution is*

$$x_i = \begin{cases} 1 & \textit{for } i = 6 \\ 0 & \textit{otherwise} \end{cases} \qquad \textit{for an expected return } \mathbf{r}^T\mathbf{x} = 12.$$

2. *The largest expected return of* 12 *is achieved not just by a single asset class, but by asset classes* A_6, A_{11} *and* A_{13}. *All of the portfolio could have been allocated to any one of these asset classes, so*

$$\begin{aligned} \mathbf{x}^1 &= (0,0,0,0,0,1,0,0,0,0,0,0,0,0)^T, \\ \mathbf{x}^2 &= (0,0,0,0,0,0,0,0,0,0,1,0,0,0)^T, \\ \mathbf{x}^3 &= (0,0,0,0,0,0,0,0,0,0,0,0,1,0)^T, \end{aligned}$$

are each solutions with expected return of 12. *In fact any convex combination*

$$\mathbf{x} = x_1\mathbf{x}^1 + x_2\mathbf{x}^2 + x_3\mathbf{x}^3 \qquad x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, \qquad x_1 + x_2 + x_3 = 1.$$

is a solution with expected return 12.

3. *The general solution can be obtained by defining*

$$\mathcal{A} = \{\, j \in 1, \ldots, n : \mu_j = \max_{i=1,\ldots,n} \mu_i \,\} \tag{8.2.3}$$

The solution set to (8.2.1) is then

$$\{\, \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum_{i \in \mathcal{A}} x_i \mathbf{e}_i \quad \textit{where} \quad x_i \geq 0 \; i \in \mathcal{A} \quad \textit{and} \quad \sum_{i \in \mathcal{A}} x_i = 1 \,\}. \tag{8.2.4}$$

Every point in this solution set has expected return μ_j *for any* $j \in \mathcal{A}$, *as all these asset classes have the same maximal return.*

Frequently a fund manager has constraints on how much of a portfolio can be allocated to a particular asset class. To maximize the expected return on the portfolio we want to

$$\begin{array}{ll} \textit{Minimize} & -\mathbf{r}^T\mathbf{x} \\ \mathbf{x} \in \mathbb{R}^n & \\ \textit{Subject to} & \mathbf{e}^T\mathbf{x} = 1 \\ & \ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u. \end{array} \tag{8.2.5}$$

Here the objective function is linear, there is one general linear equality constraint, m general linear inequality constraints, and simple lower and upper bounds on the variables. This is a linear programming problem, with some special constraint structure. In the special case when there are no general linear inequalities (i.e. not A matrix)

1. What are the possible values for ℓ and u?

2. What is the solution to the linear programming problem assuming $\mathbf{r} \geq 0$?

8.2.2 Minimizing Risk

Suppose the objective is to minimize the risk associated with the portfolio, subject to lower and upper bounds on the portion of the portfolio that can be allocated to a given asset. That is we want to minimize $\mathcal{V}[P]$. From (8.1.8) this can be done by solving

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & \mathbf{x}^T C \mathbf{x} \\ Subject\ to & \mathbf{e}^T \mathbf{x} = 1 \\ & \ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u. \end{array}$$

Here the objective is a quadratic function of the variables $\mathbf{x}$. Moreover, as the covariance matrix is positive semi-definite, the objective is also convex on $\mathbb{R}^n$. Again there is one general linear equality constraint, general linear inequality constraints, and simple lower and upper bounds on the variables, so the feasible region

$$\Omega = \{ \mathbf{x} \in \mathbb{R}^n : \mathbf{e}^T \mathbf{x} = 1,\ \ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u \}$$

is also a convex set. Thus we have a convex quadratic programming problem, and any local minimum is guaranteed to be a global minimum.

Another variation on this is to add a constraint specifying the return μ on the portfolio, and then choosing the portfolio that achieves that return and minimizes the risk. This produced the convex programming problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & \mathbf{x}^T C \mathbf{x} \\ Subject\ to & \mathbf{e}^T \mathbf{x} = 1 \\ & \mathbf{r}^T \mathbf{x} = \mu \\ & \ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u. \end{array} \tag{8.2.6}$$

The efficient frontier (see Figure 8.1.3) is produced by solving (8.2.6) for a range of values of μ. This is a *parametric programming problem* in which a series of optimization problems have to be solved as a parameter varies. One way of improving the efficiency of this process is to change the parameter μ, using the solution (and corresponding active set) from the previous value of μ as a starting point for solving the current problem.

8.2.3 Balancing Objectives

In practice you would like to balance the objectives of

1. Maximizing the return from the portfolio.
2. Minimizing the risk associated with the portfolio.

This is an example of *multi-criterion* or *multi-objective* optimization, where there is more that one objective.

Suppose there are m objectives $f_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ for $j = 1, \ldots, m$. One approach is to select weights ω_j for $j = 1, \ldots, m$ and to minimize the weighted combination

$$f(\mathbf{x}) = \sum_{j=1}^{m} \omega_j f_j(\mathbf{x})$$

of the individual objective functions. The difficulty is choosing appropriate weights ω_j. If the aim is to minimize each $f_j(\mathbf{x})$ the you could select weights satisfying

$$\omega_j \geq 0 \qquad \sum_{j=1}^{m} \omega_j = 1.$$

With the two objectives of maximizing returns and minimizing risk this leads to

$$f(\mathbf{x}) = -\omega_1 \mathbf{r}^T\mathbf{x} + \omega_2 \mathbf{x}^T C\mathbf{x}$$

where $\omega_1 \geq 0, \omega_2 \geq 0$ and $\omega_1 + \omega_2 = 1$. The restriction that the sum of the weights is 1 can be used to eliminate one weight producing the family of objectives

$$f_\omega(\mathbf{x}) = -\omega \mathbf{r}^T\mathbf{x} + (1-\omega)\mathbf{x}^T C\mathbf{x} \qquad \text{for} \quad \omega \in [0,1]. \tag{8.2.7}$$

The choice $\omega = 1$ puts all the emphasis on maximizing the expected return, while the choice $\omega = 0$ puts all the emphasis on minimizing the risk.

As there are only two competing objectives in this problem the objective can also be formulated as maximizing

$$\text{Portfolio return} - \frac{\text{Portfolio variance}}{\tau} = \mathbf{r}^T\mathbf{x} - \frac{\mathbf{x}^T C\mathbf{x}}{\tau},$$

where $\tau > 0$ is a risk tolerance parameter. For any value of $\tau > 0$ the optimization problem

$$\begin{array}{ll} Minimize & -\mathbf{r}^T\mathbf{x} + \frac{\mathbf{x}^T C\mathbf{x}}{\tau} \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & \mathbf{e}^T\mathbf{x} = 1 \\ & \ell \leq \begin{bmatrix} \mathbf{x} \\ A\mathbf{x} \end{bmatrix} \leq u. \end{array}$$

is a convex quadratic programming problem, with one general linear equality constraint, general linear inequality constraints, and simple lower and upper bounds on the variables.

In practice the simple strategy of a linear weighting to balance the objectives of maximizing return and minimizing risk may not be adequate.

8.3 Scenarios

Given a set of assets $\{1, \ldots, n\}$, the return on these assets during the planning period must be estimated. One approach is to represent the stochastic nature of the problem using scenarios. The investor tries to predict the future performance of each asset in each of a set of representative scenarios. Let the number of scenarios be s and let the return on asset i under scenario k be R_{ki} for $k = 1, \ldots, s$ and $i = 1, \ldots, n$. This gives a scenario return matrix $R \in \mathbb{R}^{s\times n}$. Additionally the probability of each scenario occurring is required. Let $\mathbf{p} \in \mathbb{R}^s$ be the scenario probability vector. Thus $\mathbf{p} \geq 0$ and $\mathbf{e}^T\mathbf{p} = 1$, where p_k is the probability of scenario k for $k = 1, \ldots, s$.

The expected return for the assets are $\mathbf{r} = R^T p \in \mathbb{R}^n$, the scenario returns for a portfolio with weights $\mathbf{x}$ are $\mathbf{z} = R\mathbf{x} \in \mathbb{R}^s$. and the expected portfolio return is $\mu = \mathbf{p}^T\ z = \mathbf{r}^T\mathbf{x} = \mathbf{p}^T R\mathbf{x}$. Define $P, D \in \mathbb{R}^{s\times s}$ and $S \in \mathbb{R}^{s\times n}$ by

$$P = \mathbf{p}\mathbf{e}^T, \qquad D = \operatorname{diag}(\mathbf{p}) \quad \text{and} \quad S = (I - P^T)R, \tag{8.3.1}$$

where $\mathbf{e} = [\,1 \quad \ldots \quad 1\,]^T \in \mathbb{R}^s$. For any vector $\mathbf{y}$ let $\mathbf{y}^j$ denote the vector with each component raised to the power j. As $\mathbf{z} - \mu\mathbf{e} = R\mathbf{x} - (\mathbf{p}^T R\mathbf{x})\mathbf{e} = (I - P^T)R\mathbf{x} = S\mathbf{x}$ the variance $\sigma^2(\mathbf{x})$ satisfies

$$\begin{aligned} \sigma^2(\mathbf{x}) &= \mathcal{E}[\,(\mathbf{z} - \mu\mathbf{e})^2\,] = \mathbf{p}^T(S\mathbf{x})^2 = \mathbf{x}^T S^T DS\mathbf{x}, && (8.3.2) \\ \nabla\sigma^2(\mathbf{x}) &= 2S^T DS\mathbf{x}, && (8.3.3) \\ \nabla^2\sigma^2(\mathbf{x}) &= 2S^T DS, && (8.3.4) \end{aligned}$$

so the covariance matrix is $C = S^T DS$.

Example 8.3.1 (Scenario data) *Let the number of asset classes be $n = 9$ and the number of scenarios*

be $s = 15$. *Let the matrix* $R \in \mathbb{R}^{s \times n}$ *of scenario returns be*

$$R = \begin{bmatrix} 12.4 & 8.6 & 1.0 & 9.1 & 5.1 & 5.7 & 4.1 & 10.6 & 8.8 \\ 12.4 & 8.6 & 10.0 & 9.1 & 5.1 & 5.7 & 4.1 & 10.6 & 8.8 \\ 12.4 & 3.0 & 10.0 & 2.0 & 5.1 & 5.7 & 4.1 & 10.6 & 8.8 \\ 12.4 & 12.0 & 1.0 & 9.1 & 5.1 & 5.7 & 4.1 & 10.6 & 8.8 \\ 10.5 & 6.0 & 8.0 & 8.0 & 10.8 & 5.5 & 4.1 & 8.0 & 9.5 \\ 5.0 & 4.0 & 4.5 & 7.0 & 10.0 & 5.5 & 4.1 & 8.0 & 9.5 \\ -2.0 & 2.0 & 1.0 & 4.0 & 15.0 & 5.0 & 4.1 & -4.0 & 13.0 \\ -7.0 & 0.0 & -1.0 & 2.0 & 15.0 & 5.0 & 4.1 & -4.0 & 13.0 \\ -10.0 & -4.0 & -6.0 & -2.0 & 20.0 & 4.5 & 4.1 & -14.0 & 17.0 \\ -17.0 & -8.0 & -10.0 & -4.0 & 20.0 & 4.5 & 4.1 & -14.0 & 17.0 \\ 20.0 & 10.0 & 16.0 & 9.5 & 8.1 & 6.0 & 4.1 & 12.0 & 8.8 \\ 16.0 & 9.0 & 4.0 & 9.5 & 5.0 & 6.5 & 4.1 & 11.0 & 7.0 \\ 16.0 & 9.0 & 4.0 & 9.5 & 5.0 & 6.5 & 4.1 & 18.0 & 7.0 \\ 25.0 & 15.0 & 10.0 & 12.0 & 0.0 & 8.0 & 4.1 & 20.0 & 3.0 \\ 18.0 & 12.0 & 8.0 & 10.0 & 0.0 & 8.0 & 4.1 & 20.0 & 3.0 \end{bmatrix}$$

and the vector $\mathbf{p} \in \mathbb{R}^s$ *of scenario probabilities be*

$$\mathbf{p} = [\ 0.15 \quad 0.12 \quad 0.10 \quad 0.03 \quad 0.08 \quad 0.10 \quad 0.03 \quad 0.05 \quad 0.01 \quad 0.01 \quad 0.05 \quad 0.09 \quad 0.07 \quad 0.06 \quad 0.0\ ]^T.$$

The expected asset returns $\mathbf{r} = R^T\mathbf{p} \in \mathbb{R}^n$ *are*

$$\mathbf{r} = [\ 11.580 \quad 7.242 \quad 5.730 \quad 7.645 \quad 6.709 \quad 5.980 \quad 4.100 \quad 10.130 \quad 8.500\ ]^T.$$

The covariance matrix $C = S^T D S$ *is*

$$C = \begin{bmatrix} 59.8476 & 28.1344 & 23.1586 & 18.7829 & -28.9722 & 4.9686 & 0 & 47.4106 & -18.0520 \\ 28.1344 & 16.8026 & 7.7733 & 12.1511 & -14.4444 & 2.6502 & 0 & 22.9777 & -9.2354 \\ 23.1586 & 7.7733 & 22.4921 & 4.4422 & -8.7126 & 1.4356 & 0 & 16.8231 & -5.8660 \\ 18.7829 & 12.1511 & 4.4422 & 10.2895 & -8.8878 & 1.6739 & 0 & 15.7692 & -6.1485 \\ -28.9722 & -14.4444 & -8.7126 & -8.8878 & 18.0050 & -2.8098 & 0 & -25.7162 & 10.1975 \\ 4.9686 & 2.6502 & 1.4356 & 1.6739 & -2.8098 & 0.6856 & 0 & 4.4756 & -2.0710 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 47.4106 & 22.9777 & 16.8231 & 15.7692 & -25.7162 & 4.4756 & 0 & 43.8171 & -16.4030 \\ -18.0520 & -9.2354 & -5.8660 & -6.1485 & 10.1975 & -2.0710 & 0 & -16.4030 & 6.9730 \end{bmatrix}.$$

The eigenvalues of the covariance matrix C *are*

$$[\ 149.8308 \quad 16.6335 \quad 6.3425 \quad 3.0275 \quad 1.9212 \quad 0.3347 \quad 0.8025 \quad 0.0199 \quad 0\ ]^T.$$

The eigenvalues are all non-negative confirming the fact that C *is positive semi-definite. However* C *is not positive definite as there is one zero eigenvalue. This occurs in this example as asset class 7 (a short term bank deposit) has the same return in all scenarios.*

Solving the problem (8.2.6) for portfolio returns μ *in the range* $[4.1, 11.58]$ *produces the asset allocation in Figure 8.3.1 and the efficient frontier in Figure 8.3.2. In Figure 8.3.2 the portfolio risk is measured by the standard deviation* $\sqrt{\mathbf{x}^T C \mathbf{x}}$ *of the portfolio.*

8.4 Other measures of risk

The variance model measures risk by the amount the return differs from the expected. Typically an investor is not concerned if their return is higher than expected, only if their return is *lower* than expected. Several asymmetric measures of risk have been studied (see [Kan82, Kin93] for example). This section considers two such measures of risk: the semi-variance and the skewness.

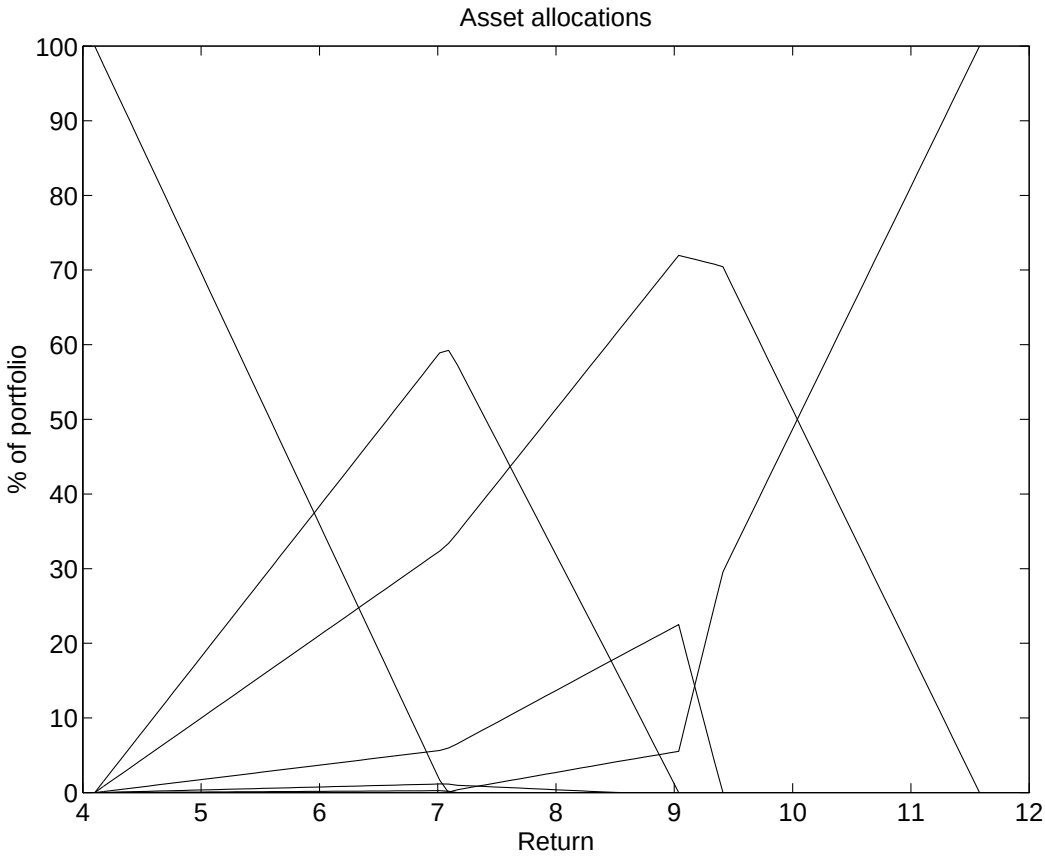


Figure 8.3.1: Asset allocations for different portfolio returns μ

8.4.1 A Semi-variance Model

The downside-risk approach to investment decisions [Har91] focuses on the variability of returns below a specified target or benchmark. Let P be a random variable representing the portfolio return. One measure of the down-side risk of the portfolio is the *semi-variance* defined by

$$\varphi(\mathbf{x}) = \mathcal{E}[\min(0, P-\mu)^2] = \mathbf{p}^T(\min(0, S\mathbf{x})^2). \tag{8.4.1}$$

The semi-variance $\varphi(\mathbf{x})$ is once continuously differentiable, but not necessarily twice continuously differentiable. It is a convex piecewise quadratic function, where each quadratic piece has a positive semi-definite Hessian $G(x)$.

Let $\widehat{D}(\mathbf{x}) \in \mathbb{R}^{s\times s}$ be the diagonal matrix defined by

$$\widehat{D}_{kk}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{e}_k^T S\mathbf{x} > 0 \\ 0 \text{ or } p_k & \text{if } \mathbf{e}_k^T S\mathbf{x} = 0 \\ p_k & \text{if } \mathbf{e}_k^T S\mathbf{x} < 0. \end{cases} \tag{8.4.2}$$

Then

$$\varphi(\mathbf{x}) = \mathbf{x}^T S^T \widehat{D}(\mathbf{x}) S\mathbf{x}, \tag{8.4.3}$$

$$\nabla\varphi(\mathbf{x}) = 2S^T \widehat{D}(\mathbf{x}) Sx, \tag{8.4.4}$$

$$G(\mathbf{x}) = 2S^T \widehat{D}(\mathbf{x}) S. \tag{8.4.5}$$

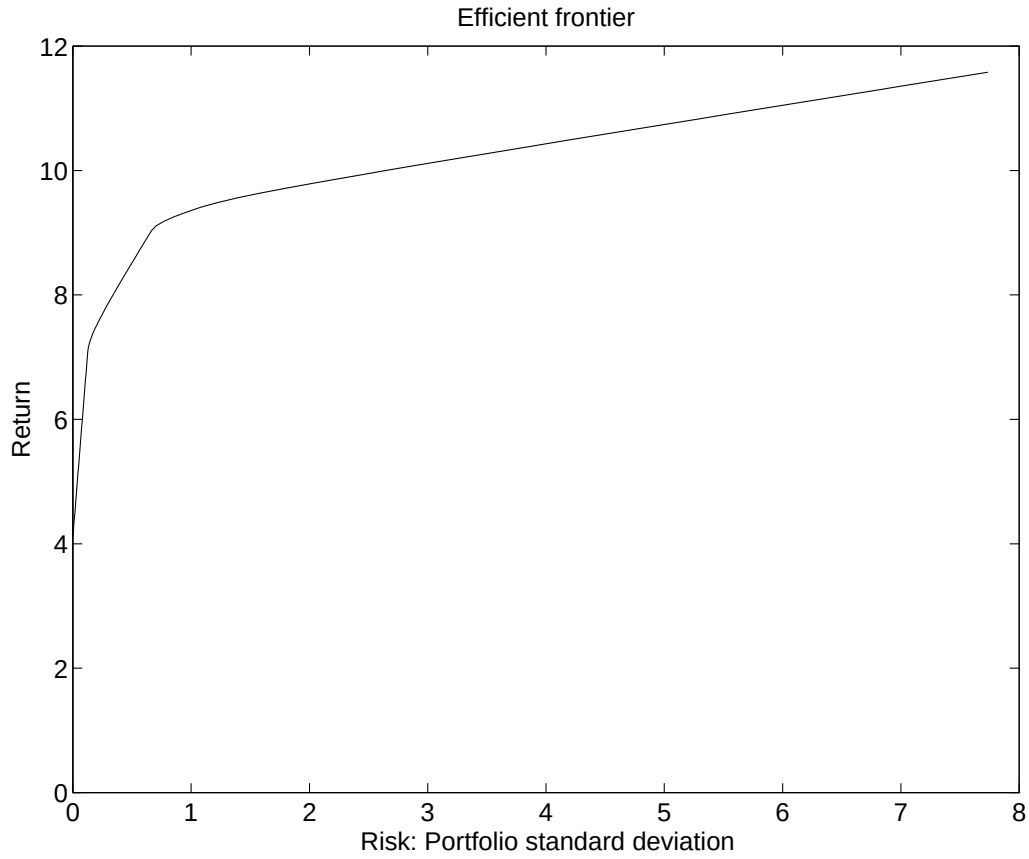


Figure 8.3.2: Efficient frontier

where $G(\mathbf{x})$ is an element of the generalized Hessian. If $\mathbf{e}_k^T S\mathbf{x} \neq 0$ for $k = 1, \ldots, s$ then $\mathbf{x}$ is in the interior of a quadratic piece of the semi-variance.

The semi-variance $\varphi(\mathbf{x})$ can be used instead of the variance $\sigma^2(\mathbf{x})$ in (8.2.6). As the semi-variance is a convex function, the resulting problem is a convex programming problem and so any local minimizer is a global minimizer. As in the mean – variance model, an efficient frontier for the semi variance model can be calculated by varying the expected return μ.

8.4.2 Skewness Model

Other attempts to model the asymmetric nature of risk are based on *maximizing* the skewness

$$\phi(\mathbf{x}) = \frac{\kappa(\mathbf{x})}{\sigma^{3/2}(\mathbf{x})}, \tag{8.4.6}$$

where

$$\kappa(\mathbf{x}) = \mathcal{E}[\,(P-\mu)^3\,] = p^T(S\mathbf{x})^3. \tag{8.4.7}$$

The skewness function $\phi(\mathbf{x})$ is the third moment of the returns scaled by the variance raised to the power $\frac{3}{2}$. A positively skewed portfolio is more likely to have a return exceeding the expected value, while a symmetric return distribution with equal probability of getting returns lower or higher than expected will have a zero skewness. Unlike semi-variance, the skewness function is C^∞, as long as the variance is non-zero. Let $E(\mathbf{x}) = \mathrm{diag}(\,DS\mathbf{x}\,)$. Then

$$\nabla\kappa(\mathbf{x}) \quad = \quad 3S^T D(S\mathbf{x})^2$$

$$\nabla^2\kappa(\mathbf{x}) = 6S^T E(\mathbf{x})S$$

The gradient and the Hessian of $\phi(\mathbf{x})$ are

$$\nabla\phi(\mathbf{x}) = \frac{3}{\sigma^3(\mathbf{x})}S^T D(S\mathbf{x})^2 - \frac{3\kappa(\mathbf{x})}{\sigma^5(\mathbf{x})}S^T DS\mathbf{x} \tag{8.4.8}$$

$$\begin{aligned}\nabla^2\phi(x) &= \frac{6}{\sigma^3(\mathbf{x})}S^T E(\mathbf{x})S + \frac{15\kappa(\mathbf{x})}{\sigma^7(\mathbf{x})}S^T DS\mathbf{x}\mathbf{x}^T S^T DS \\ &\quad -\frac{3}{\sigma^5(\mathbf{x})}\left(\kappa(\mathbf{x})S^T DS + 3S^T D(S\mathbf{x}(\mathbf{x}^T S^T)^2 + (S\mathbf{x})^2\mathbf{x}^T S^T)DS\right).\end{aligned} \tag{8.4.9}$$

The skewness model can be formulated as maximizing $\phi(\mathbf{x})$ subject to the same set of constraints as in the mean variance model (8.2.6). The variance is not restricted as a constraint specifying the value of variance will lead to a quadratic constraint which is much harder to solve than a linearly constrained problem. As $\phi(\mathbf{x})$ is the third moment $\kappa(\mathbf{x})$ divided by the variance $\sigma^2(\mathbf{x})$ raised to the power $\frac{3}{2}$, maximizing the skewness tends to maximize the third moment while minimizing the variance.

An alternative skewness model has been proposed [KY91, KSY93, FT93] in which the absolute deviation of the return from the mean is used as a surrogate for the variance. Instead of maximizing the skewness their model maximizes the third moment, which simplifies the objective function. A quadratic constraint on the variance is replaced by a piecewise linear constraint controlling the absolute deviation of the return. Additional variables are then used to convert the absolute deviation into a number of linear constraints, which increases the size of the problem substantially.

Incorporating the skewness recognises the asymmetric nature of return and provides a better approximation than the mean variance model. However, as the skewness function is not convex, the problem may have several different local minima.

8.5 Exercises

1. Prove the following results

$$\mathcal{V}[Y] = \mathcal{E}[Y^2] - \mu^2 \tag{8.5.1}$$

$$Cov[Y,Z] = \mathcal{E}[YZ] - \mathcal{E}[Y]\mathcal{E}[Z] \tag{8.5.2}$$

$$\mathcal{V}[W] = \alpha^2\mathcal{V}[Y] + \beta^2\mathcal{V}[Z] + 2\alpha\beta Cov[Y,Z]. \tag{8.5.3}$$

where $W = \alpha Y + \beta Z$ and $\alpha, \beta \in \mathbb{R}$.

2. The annualized monthly returns illustrated in Figure B.1.3 are given in Table 8.5.1. Suppose that

Asset	Annualized monthly return %											
A_1	5	15	20	10	0	-10	-20	0	8	16	4	-12
A_2	3	12	10	8	4	-5	-10	-5	4	12	5	-2
A_3	0	-10	-5	8	10	15	12	5	-8	-6	0	15

Table 8.5.1: Annualized monthly returns for 3 assets

each value in this table has equal probability (is this likely to be a reasonable assumption?).

(a) Show that each asset has the same expected return, but different risk.

(b) Calculate the covariance matrix for these three assets.

(c) You want to minimize the risk associated with a portfolio of 2 assets chosen from A_1, A_2 and A_3.

i. What is the best solution if A_1 and A_2 are chosen?

ii. What is the best solution if A_1 and A_3 are chosen?

3. The following data from AMP investments is derived from broad trends based on historical data over various periods (they are NOT AMP Investments forecasts). The major asset classes are: Australian Shares (A); Direct Property (P); Australian Fixed Interest (F); Cash Equivalents (C) and International Shares (I). The returns are before tax and assume all income is re-invested. There are also some typical constraints that are applied to the portfolio weights.

Asset Class	Expected Return	Standard Deviation	Constraints on % of portfolio
A	12.5	20	25 - 55
P	10.5	8	5 - 20
F	7.5	5	10 - 25
C	7.0	1	3 - 20
I	11.5	16	10 - 35

Table 8.5.2: Data for AMP asset classes

The correlation matrix for these asset classes is

	A	P	F	C	I
A	1	0.05	0.40	-0.10	0.50
P		1	0.11	-0.02	0.09
F			1	-0.12	0.14
C				1	0.02
I					1

(a) Find the portfolio that maximizes the expected return from the portfolio.

(b) Find the portfolio that minimizes the risk of the portfolio.

(c) Find the portfolio that gives a 80% weighting to maximizing the return and a 20% weighting to minimizing the risk.

(d) Find the portfolio that gives a 20% weighting to maximizing the return and a 80% weighting to minimizing the risk.

8.6 References

1. C. F. Garvin, *Financial Toolbox User's Guide: For use with MATLAB*, (The MathWorks Inc., 1995).
2. Z. Bodie, A. Kane and A. J. Marcus, *Investments*, (3rd edition, Irwin, 1996).
3. H. Markowitz, "Portfolio selection", *Journal of Finance* **7**, 1 (March 1952) 77-91.
4. H. Markowitz, *Portfolio Selection: Efficient diversification of investments*, (Yale University Press, 1959).
5. W. Mendenhall, R. L. Scheaffer and D. D. Wackerly, *Mathematical Statistics with Applications*, (Duxbury Press, 1981).
6. F. K. Reilly, *Investment Analysis and Portfolio Management*, (Dryden Press, 3rd Edition, 1989).
7. W. F. Sharpe, *Portfolio Theory and Capital Markets*, (McGraw-Hill, N.Y. 1970)
8. A. Rudd and H. K. Clasing, *Modern Portfolio Theory: The principles of investment management*, (2nd Edition, Andrew Rudd, 1988).

Chapter 9

Approximation Problems [H]

9.1 Function Approximation

9.1.1 Introduction

Let $y(t)$ be a function of a real variable t (for example t could be time or temperature). Consider approximating y over a set Γ (typically an interval) by a function $\phi(t)$, where ϕ is to be chosen to produce the 'best' approximation. This requires a definition of 'best'; that is a measure of the distance between the functions y and ϕ on the set Γ. The functions y and ϕ lie in a function space (for example the space $C^k(\Gamma)$ of functions defined on Γ and with $k \geq 0$ continuous derivatives, or $L_2(\Gamma)$ of square integrable functions on Γ). Possible measures of the distance between two functions include the norms (norms on function spaces, not $\mathbb{R}^n$). For example

$$||\phi - y||_1 = \int_\Gamma |\phi(t) - y(t)| \, dt, \tag{9.1.1}$$

$$||\phi - y||_2 = \left[\int_\Gamma |\phi(t) - y(t)|^2 \, dt\right]^{\frac{1}{2}}, \tag{9.1.2}$$

$$||\phi - y||_\infty = \max_{t \in \Gamma} |\phi(t) - y(t)|. \tag{9.1.3}$$

Often there are other characteristics of the function ϕ which may be desirable (or undesirable) and so should be included in the definition of best. For example the second derivative $\phi''(t)$ is a measure of the curvature of ϕ. To minimize the curvature, and hence the number of oscillations in ϕ, the term

$$\int_\Gamma |\phi''(t)|^2 \, dt = ||\phi''||_2^2 \tag{9.1.4}$$

can be included in the definition of 'best'. This is referred to as *regularization*. An objective that attempts to balance the tasks of approximating y and minimizing the curvature in ϕ is

$$f_\tau(\phi) = ||\phi - y||_2^2 + \tau||\phi''||_2^2 \qquad \tau \geq 0 \in \mathbb{R}. \tag{9.1.5}$$

Large values of τ emphasize the objective of minimizing the oscillations in ϕ. However the difficulty is finding an *automatic* way of choosing the regularization parameter τ.

Example 9.1.1 *The error function*

$$erf\ (t) = \frac{2}{\pi} \int_0^t e^{-s^2} \, ds \tag{9.1.6}$$

is sketched in the first plot in Figure 9.1.1.

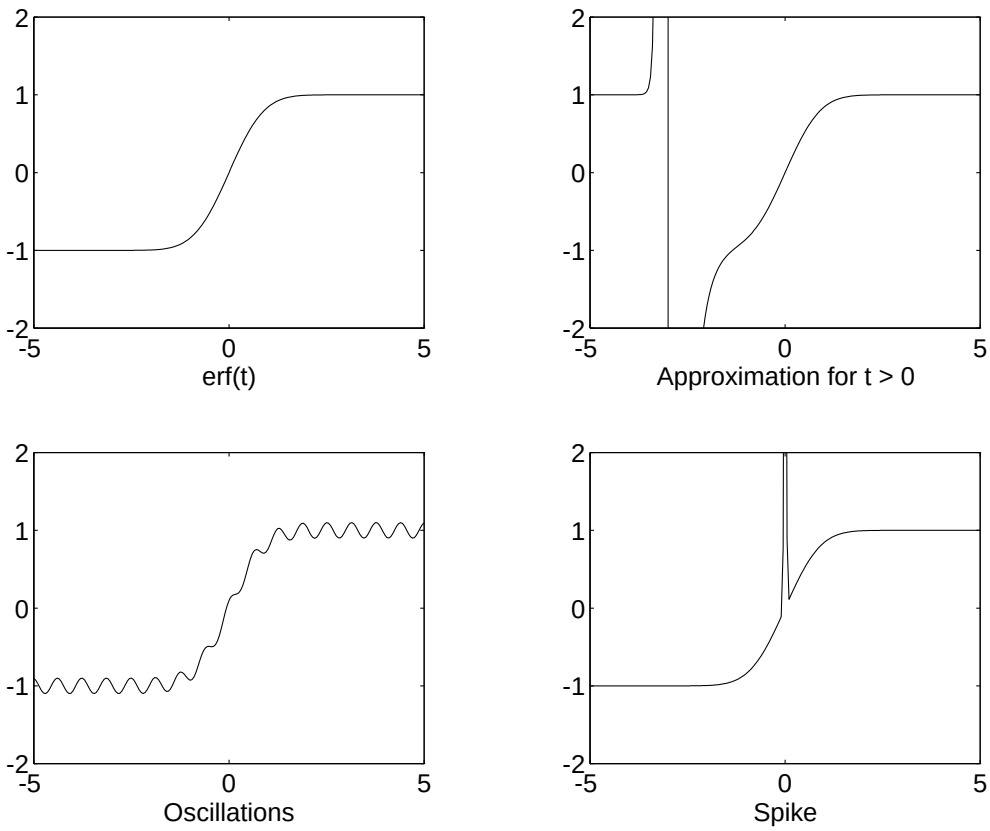


Figure 9.1.1: Approximations to the error function

The error function is related to the normal probability density function and can be used to calculate probabilities for a random variable Y which is normally distributed

$$\begin{aligned} P(Y \le t) &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{(s-\mu)^2}{2\sigma^2}}\, ds \\ &= \frac{1}{2}(1 + erf\ (\frac{t-\mu}{\sigma\sqrt{2}})). \end{aligned}$$

Unfortunately (9.1.6) cannot be evaluated analytically, and approximations or tables must be used. One such approximation from Abramowitz and Stegun (1972) is

$$erf\ (t) = \phi(t) + \epsilon(t) \qquad t \in \Gamma \equiv [0, \infty)$$

where

$$\phi(t) = 1 - (a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 + a_5 r^5)e^{-t^2} \tag{9.1.7}$$

$$r = \frac{1}{1+pt} \qquad p = 0.3275911$$

and

$$\begin{aligned} a_1 &= 0.254829592 \qquad a_2 = -0.284496736 \\ a_3 &= 1.421413741 \qquad a_4 = -1.453152027 \\ a_5 &= 1.061405429. \end{aligned}$$

This approximation is for $\Gamma = [0, \infty)$*, and the error* $\epsilon(t)$ *satisfies*

$$|\epsilon(t)| \leq 1.5 \times 10^{-7} \quad \forall t \in \Gamma. \tag{9.1.8}$$

The approximating function (9.1.7) is illustrated in the second plot in Figure 9.1.1. Note that the approximation bears little relation to $erf\ (t)$ *when* $t < 0$*. Equation (9.1.8) implies* $||\phi - erf\ ||_\infty \leq 1.5 \times 10^{-7}$ *for* $\Gamma = [0, \infty)$*.*

Norm	Rational approximation on $[0, 5]$	Oscillating approximation	Approximation with a spike
$\|\|\phi - erf\ \|\|_1$	2.6×10^{-7}	0.63	0.56
$\|\|\phi - erf\ \|\|_2$	1.6×10^{-7}	0.22	1.99
$\|\|\phi - erf\ \|\|_\infty$	1.4×10^{-7}	0.10	10.00

Table 9.1.1: Norms of errors in approximating the error function

The norms of the errors for the approximating functions in Figure 9.1.1 are given in Table 9.1.1. The third plot in Figure 9.1.1 gives an example of an oscillating approximation to $erf\ (t)$*, or an example where* $erf\ (t)$ *is approximating an oscillating function. In this situation the* L_1 *norm can be relatively large, while the* L_∞ *norm is relatively small. The final plot is an example of an approximation that would give an error with a low* L_1 *or* L_2 *norm, but the* L_∞ *norm gives a large error (just make the spike arbitrarily narrow and high).*

Another possibility is to include constraints on the permissible approximating functions ϕ. For example

1. Non-negativity constraints: $\phi(t) \geq 0 \quad t \in \Gamma$.
2. Monotonicity constraints: $\phi'(t) \geq 0 \quad t \in \Gamma$.
3. Convexity constraints: $\phi''(t) \geq 0 \quad t \in \Gamma$.

Other objective functions are obtained by including a *weight function* which gives different emphasis to different parts of the set Γ. Let $w(t)$ be a nonnegative weight function defined on Γ such that

1. $$\int_\Gamma \phi(t)w(t)\ dt \text{ is integrable and finite for all } \phi \in C[\Gamma]$$
2. If for some continuous nonnegative function $y(t)$
$$\int_\Gamma w(t)y(t)\ dt = 0$$
then the function $y(t) \equiv 0$ on Γ.

Common weight functions are

1. $w(t) \equiv 1 \qquad t \in \Gamma$
2. $w(t) = 1/\sqrt{1-t^2}$ on $\Gamma = [-1, 1]$
3. $w(t) = e^{-t}$ on $\Gamma = [0, \infty)$
4. $w(t) = e^{-t^2}$ on $\Gamma = (-\infty, \infty)$.

9.1.2 Inner products

An *inner product* $\langle y, z \rangle$ is a map from a vector space to $\mathbb{R}$ satisfying

1. $\langle \alpha y, z \rangle = \langle y, \alpha z \rangle = \alpha \langle y, z \rangle \qquad \alpha \in \mathbb{R}$
2. (a) $\langle y_1 + y_2, z \rangle = \langle y_1, z \rangle + \langle y_2, z \rangle$
   (b) $\langle y, z_1 + z_2 \rangle = \langle y, z_1 \rangle + \langle y, z_2 \rangle$
3. $\langle y, z \rangle = \langle z, y \rangle$
4. $\langle y, y \rangle \geq 0$ for all y, and $\langle y, y \rangle = 0$ if and only if $y(t) = 0$ for all $t \in \Gamma$.

Any inner product produces a 2-norm or Euclidean norm defined by

$$||y||_2 = [\langle y, y \rangle]^{\frac{1}{2}} . \tag{9.1.9}$$

It is easily verified that if $y, z \in C(\Gamma)$ then

$$\langle y, z \rangle = \int_\Gamma w(t) y(t) z(t) \, dt \tag{9.1.10}$$

satisfies the properties which define a general inner product. Thus a general least squares problem is to find the function ϕ that minimizes

$$||\phi - y||_2^2 = \int_\Gamma w(t) [\phi(t) - y(t)]^2 \, dt \tag{9.1.11}$$

over all functions ϕ in a certain class (for example polynomials of degree $< n$).

9.1.3 Linear least squares

The physical problem which produced the function $y(t)$ may suggest a particular parametric form for the function $\phi(t)$, so it can be written as $\phi(\mathbf{x}; t)$ where $\mathbf{x} \in \mathbb{R}^n$ is a vector of parameters defining a class of approximating functions. Of particular importance is the case when $\phi(\mathbf{x}; t)$ is *linear* in the parameters $\mathbf{x}$, so

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n} x_j \psi_j(t). \tag{9.1.12}$$

Here the approximation is coming from the linear space spanned by the functions $\psi_j(t)$ for $j = 1, \ldots, n$. It is sensible to restrict attention to sets of functions ψ_j $j = 1, \ldots, n$ which are linearly independent, so they form a basis for this linear space.

The linear least squares problem is then to minimize

$$\begin{aligned}
||\phi - y||_2^2 &= \langle \phi - y, \phi - y \rangle \\
&= \langle \phi, \phi \rangle - 2\langle \phi, y \rangle + \langle y, y \rangle \\
&= \langle \sum_{i=1}^{n} x_i \psi_i, \sum_{j=1}^{n} x_j \psi_j \rangle - 2\langle \sum_{j=1}^{n} x_j \psi_j, y \rangle + \langle y, y \rangle \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \langle \psi_i, \psi_j \rangle - 2 \sum_{j=1}^{n} x_j \langle \psi_j, y \rangle + \langle y, y \rangle \\
&= \mathbf{x}^T \Psi \mathbf{x} - 2\mathbf{x}^T \mathbf{d} + \langle y, y \rangle
\end{aligned} \tag{9.1.13}$$

where the n by n symmetric matrix Ψ is defined by

$$\Psi_{ij} = \langle \psi_i, \psi_j \rangle \qquad i = 1, \ldots, n, \; j = 1, \ldots, n \tag{9.1.14}$$

and

$$\mathbf{d}_j = \langle \psi_j, y \rangle \qquad j = 1, \ldots, n. \tag{9.1.15}$$

As

$$\langle \phi, \phi \rangle = \mathbf{x}^T \Psi \mathbf{x}$$

the matrix Ψ is nonnegative definite. Moreover if the functions ψ_j for $j = 1, \ldots, n$ are linearly independent then the matrix Ψ is positive definite. As (9.1.13) is a strictly convex quadratic function the solution of the linear least squares problem is given by

$$\Psi \mathbf{x} = \mathbf{d}. \tag{9.1.16}$$

When Ψ is positive definite this can be written as

$$\mathbf{x}^* = \Psi^{-1} \mathbf{d}. \tag{9.1.17}$$

9.1.4 Basis functions

Particular forms of the approximating function include

1. Monomial approximations where

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n} x_j t^{j-1}$$

2. Rational approximations where

$$\phi(\mathbf{x}; t) = \frac{\sum_{i=1}^{\ell} x_i t^{i-1}}{\sum_{j=1}^{n-\ell} x_{\ell+j} t^{j-1}}$$

3. Fourier series where n is even and

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n/2} x_j \cos(jt) + x_{n/2+j} \sin(jt)$$

4. Trigonometric approximations where n is even and

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n/2} x_j \cos(2\pi x_{n/2+j} t)$$

5. Exponential approximations where n is even and

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n/2} x_j e^{2\pi x_{n/2+j} t}$$

6. Gaussian approximations where n is a multiple of 3 and

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n/3} x_j e^{x_{n/3+j}(t - x_{2n/3+j})^2}$$

Of the above examples only the monomial approximation problem and the Fourier series are linear with

$$\psi_j(t) = t^{j-1} \qquad j = 1, \ldots, n, \tag{9.1.18}$$

and

$$\psi_j(t) = \cos(jt) \quad \psi_{n/2+j}(t) = \sin(jt) \qquad j = 1, \ldots, n/2 \tag{9.1.19}$$

respectively. The other examples include nonlinear functions of the parameters $\mathbf{x}$.

Example 9.1.2 *The monomial basis functions (9.1.18) on* $\Gamma = [0,1]$ *produce*

$$\Psi_{ij} = \langle \psi_i, \psi_j \rangle = \int_0^1 t^{i-1} t^{j-1}\, dt = \frac{1}{i+j-1} \qquad i,j = 1,\ldots,n.$$

Thus Ψ *is the notorious Hilbert matrix, which is known to be very ill-conditioned for solving the linear system. Table 9.1.2 illustrates how rapidly the condition number of the Hilbert matrix grows with the size*

Dimension n	Condition number κ
2	1.9×10^1
4	1.6×10^4
6	1.5×10^7
8	1.5×10^{10}
10	1.6×10^{13}
12	1.7×10^{16}

Table 9.1.2: Condition numbers for Hilbert matrices

of the matrix. For dimensions > 10 *it is impossible to obtain any useful information (even in double precision), as solving the linear system* $\Psi\mathbf{x} = \mathbf{d}$ *amplifies the rounding errors inherent in storing the data. Further details on the sensitivity of linear systems can be found in Section A.8.3 of the Appendix.*

9.1.5 Orthogonal functions

The functions $\phi_j(t)$ $j = 1,\ldots,n$ are *orthogonal* with respect to the inner product iff

$$\langle \phi_i, \phi_j \rangle = 0 \qquad i \neq j. \tag{9.1.20}$$

Orthogonal functions produce an explicit solution (so a system of linear equations does not need to be solved) to the linear least squares approximation problem. From equation (9.1.14) and (9.1.20)

$$\Psi = \mathrm{diag}(\langle \psi_1, \psi_1 \rangle, \ldots, \langle \psi_n, \psi_n \rangle) \tag{9.1.21}$$

so (assuming the basis functions ψ_j are linearly independent) the solution (9.1.17) is

$$x_j = \frac{d_j}{\Psi_{jj}} = \frac{\langle \psi_j, y \rangle}{\langle \psi_j, \psi_j \rangle} \qquad j = 1,\ldots,n. \tag{9.1.22}$$

The main task is then the evaluation of the inner products $\langle \psi_j, y \rangle$ and $\langle \psi_j, \psi_j \rangle$ for $j = 1,\ldots,n$.

Using the monomials (9.1.18) as basis functions for polynomial approximation is not particularly good because they are nearly linearly dependent, and are not orthogonal. There are many orthogonal polynomials, including Legendre polynomials, Chebyshev polynomials, and the Laguerre polynomials. For example the Chebyshev polynomials for which $\Gamma = [-1,1]$ and $w(t) = 1/\sqrt{1-t^2}$ are given by

$$T_j(t) = \cos(j \cos^{-1} t) \qquad j \geq 0.$$

The Chebyshev polynomials satisfy the recurrence relation

$$T_{j+1}(t) = 2tT_j(t) - T_{j-1}(t) \qquad j \geq 1$$

and

$$T_0(t) = 1, \qquad T_1(t) = t.$$

The advantages and use of orthogonal polynomials are covered more extensively in other courses.

When approximating functions the use of the L_∞ norm (9.1.3) is important as it gives the maximum error between the approximating function and y over the set Γ. There is an extensive theory characterizing the best linear L_∞ approximations and algorithms for calculating them (e.g. the Remez algorithms discussed in Powell (1981) or Watson (1980)).

9.1.6 Exercises

1. Consider the regularized linear least squares problem (9.1.5) where ϕ satisfies (9.1.12). Let

$$\Lambda_{ij} = \langle \psi_i'', \psi_j'' \rangle.$$

   (a) Find the solution to this problem in terms of the matrices Ψ defined by (9.1.14) and Λ, and the regularization parameter τ.

   (b) For the monomial basis functions (9.1.18) on $\Gamma = [0, 1]$ with weight function $w \equiv 1$.

      i. Calculate the n by n matrix Λ.

      ii. For $n = 5$ calculate the best regularized approximation for $\tau = 0$ and $\tau = 10$ to

$$y(t) = \begin{cases} 0 & \text{if} t < 0.5; \\ 1 & \text{if} t \geq 0.5. \end{cases}$$

      iii. What happens to the best regularized solution as $\tau \to \infty$?

9.2 Data Fitting

In many applications the value of the function $y(t)$ is only measured at certain discrete values of t (for example the value may only be measured at certain times). Thus at a finite set of values t_i for $i = 1, \ldots, m$ the values $y_i \equiv y(t_i)$ are obtained. The problem is to find a function of $\phi(t)$ which approximates the data values y_i for $i = 1, \ldots, m$.

One way of doing this is to minimize some measure of the distance between the data values y_i and the function values ϕ_i (the approximation function $\phi(t_i)$ evaluated at the sample point t_i). Let $\mathbf{y} \in \mathbb{R}^m$ be the vector of data values and let $\phi \in \mathbb{R}^m$ be the vector of approximating function values $\phi_i \equiv \phi(t_i)$ for $i = 1, \ldots, m$. Any vector norm

$$||\phi - \mathbf{y}||$$

provides a measure of the distance between ϕ and y.

Again the physical problem which produced the function $y(t)$ may suggest a particular parametric form for the function $\phi(t)$, so it can be written as $\phi(\mathbf{x}; t)$ where $\mathbf{x} \in \mathbb{R}^n$ is a vector of parameters defining a class of approximating functions. Frequently the number of data values m is far larger than the number of parameters n in the model.

Of particular importance is the case when $\phi(\mathbf{x}; t)$ is *linear* in the parameters $\mathbf{x}$ so

$$\phi(\mathbf{x}; t) = \sum_{j=1}^{n} x_j \psi_j(t).$$

Then

$$\phi(\mathbf{x}; t_i) = \sum_{j=1} x_j \psi_j(t_i) \qquad i = 1, \ldots, m, \; j = 1, \ldots, n$$

so

$$\phi = X\mathbf{x} \tag{9.2.1}$$

where $X \in \mathbb{R}^{m \times n}$ is defined by

$$X_{ij} = \psi_j(t_i) \qquad i = 1, \ldots, m, \; j = 1, \ldots, n. \tag{9.2.2}$$

Each row of X corresponds to a different data point, while each column of X corresponds to a different variable. The discrete approximation problem is then

$$\begin{array}{ll} Minimize & ||X\mathbf{x} - \mathbf{y}|| \\ \mathbf{x} \in \mathbb{R}^n & \end{array} \tag{9.2.3}$$

Frequently this is posed in terms of minimizing a norm of the *residual*

$$\mathbf{r} = X\mathbf{x} - \mathbf{y}. \tag{9.2.4}$$

9.2.1 Linear Least Squares

The linear least squares problem corresponds to minimizing

$$\begin{aligned} ||X\mathbf{x}-\mathbf{y}||_2^2 &= (X\mathbf{x}-\mathbf{y})^T(X\mathbf{x}-\mathbf{y}) \\ &= \mathbf{x}^T X^T X\mathbf{x} - 2\mathbf{y}^T X\mathbf{x} + \mathbf{y}^T\mathbf{y}. \end{aligned}$$

This is a quadratic function with gradient and Hessian given by

$$g = 2X^T X\mathbf{x} - 2X^T\mathbf{y} \qquad G = 2X^T X. \tag{9.2.5}$$

The Hessian is immediately non-negative definite. If the approximating functions ψ_j are linearly independent and the sample points t_i are distinct, then the matrix X has full rank, so $X^T X$ is positive definite. The solution of the linear least squares problem is then given by the *normal equations*

$$X^T X\mathbf{x}^* = X^T\mathbf{y}. \tag{9.2.6}$$

When X has full rank the matrix $X^T X$ is nonsingular, so this can be written as

$$\mathbf{x}^* = (X^T X)^{-1} X^T\mathbf{y}. \tag{9.2.7}$$

The *standard error*

$$\mathbf{r}^{*T}\mathbf{r}^* = ||\mathbf{r}^*||_2^2 = ||X\mathbf{x}^*-\mathbf{y}||_2^2 \tag{9.2.8}$$

is used as a measure of the 'goodness of the fit'. The matrix

$$X^+ = (X^T X)^{-1} X^T \tag{9.2.9}$$

is called the *pseudo inverse* or *generalized inverse*. Although it is very convenient to write the solution in the form (9.2.7), this is a numerically bad way of solving the linear system (9.2.6) as forming $X^T X$ increases the effects of rounding errors. The use of an orthogonal factorization to solve the linear least squares problem is discussed in Section A.8.4 (Over-determined linear systems) in the Appendix.

One important justification of least squares approximation is its statistical significance. Suppose the data values $y_i,\ i = 1,\ldots,m$ have the form

$$y_i = \bar{y}_i + \epsilon_i \qquad i = 1,\ldots,m \tag{9.2.10}$$

where $\bar{y}_i$ are the underlying data values and the errors ϵ_i are normally distributed random variables with

$$E[\epsilon_i] = 0, \qquad C_{ij} = E[\,\epsilon_i\epsilon_j\,] \quad i,j = 1,\ldots,m. \tag{9.2.11}$$

If the covariance matrix C is positive definite and X has full rank then the *generalized least squares* problem

$$\begin{array}{ll} Minimize & \mathbf{r}^T C^{-1}\mathbf{r} \\ \mathbf{x}\in\mathbb{R}^n & \\ Subject\ to & \mathbf{r} = X\mathbf{x}-\mathbf{y} \end{array} \tag{9.2.12}$$

has the solution

$$\mathbf{x}^* = (X^T C^{-1} X)^{-1} X^T C^{-1}\mathbf{y}. \tag{9.2.13}$$

This can be shown to be the minimum variance unbiased estimator, and the estimator which maximizes the likelihood when the errors are normal.

The case when $C = \sigma^2 I$ is the *ordinary least squares problem*. This assumes the errors ϵ_i in (9.2.10) are independent with mean 0 and variance σ^2. Then the covariances between the components of the estimator $\mathbf{x}^*$ are

$$Cov[\,x_i^*, x_j^*\,] = \sigma^2[(X^T X)^{-1}]_{ij}. \tag{9.2.14}$$

If the variance σ^2 of the errors is unknown it can be estimated by

$$E[\,\sigma^2\,] = \left(\frac{1}{m-n}\right)||\mathbf{r}^*||_2^2. \tag{9.2.15}$$

9.2.2 Linear ℓ_1

The Linear ℓ_1 problem is

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \quad ||X\mathbf{x} - \mathbf{y}||_1 \tag{9.2.16}$$

Let

$$\mathbf{r} = X\mathbf{x} - \mathbf{y}$$

so (9.2.16) corresponds to minimizing

$$||\mathbf{r}||_1 = \sum_{i=1}^{m} |r_i|.$$

A common strategy when dealing with absolute values is to define additional variables

$$r_i^+ = \max\{\, 0, r_i \,\} \tag{9.2.17}$$

$$r_i^- = \max\{\, 0, -r_i \,\} \tag{9.2.18}$$

corresponding to the positive and negative parts of r_i respectively. Then

$$r_i^+ \geq 0, \quad r_i^- \geq 0, \quad r_i = r_i^+ - r_i^-, \quad |r_i| = r_i^+ + r_i^- \tag{9.2.19}$$

and the *complementarity condition*

$$\mathbf{r}^{+^T}\mathbf{r}^- = 0 \tag{9.2.20}$$

is satisfied. As $\mathbf{r}^+ \geq 0$ and $\mathbf{r}^- \geq 0$ the complementarity condition (9.2.20) implies that at least one of r_i^+ and r_i^- must be zero for $i = 1, \ldots, m$.

This approach leads to the problem

$$\underset{\mathbf{x} \in \mathbb{R}^n,\, \mathbf{u}, \mathbf{v} \in \mathbb{R}^m}{Minimize} \quad \mathbf{e}^T\mathbf{u} + \mathbf{e}^T\mathbf{v} \tag{9.2.21}$$

$$Subject\ to \quad \begin{bmatrix} X & -I & I \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \mathbf{y}$$

$$\mathbf{u} \geq 0, \ \mathbf{v} \geq 0$$

$$\mathbf{u}^T\mathbf{v} = 0. \tag{9.2.22}$$

If the complementarity condition (9.2.22) is ignored this is a linear programming problem in $n + 2m$ variables $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{v}$, and with m general linear equality constraints. Fortunately it can be shown that the solution of a linear programming problem is attained at a basic feasible solution, and that any basic feasible solution of the linear programming problem automatically satisfies the complementarity problem.

If $m >> n$ then (9.2.21) is a much larger problem than the original ℓ_1 problem in n variables. Special methods that exploit the structure of the linear programming problem corresponding to (9.2.21), or direct descent methods provide more efficient ways of solving an linear ℓ_1 problem. Compared to the linear least squares problem the ℓ_1 problem has the disadvantage that a closed form solution is not available. However the ℓ_1 estimator is much less sensitive to outliers (wild data values) so in many situations (for example when the errors are unlikely to be normally distributed) the ℓ_1 estimator obtained by minimizing (9.2.16) provides a more robust estimator.

9.2.3 Linear ℓ_∞

The Linear ℓ_∞ problem is

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \quad \|X\mathbf{x} - \mathbf{y}\|_\infty \tag{9.2.23}$$

Let

$$\mathbf{r} = X\mathbf{x} - \mathbf{y},$$

so (9.2.23) corresponds to minimizing

$$||\mathbf{r}||_\infty = \max_{i=1,\ldots,m} |r_i| = \max_{i=1,\ldots,m} \max\{\, r_i, -r_i \,\}.$$

Consider adding another variable $\nu = ||\mathbf{r}||_\infty$. This implies that

$$\nu \geq r_i, \quad \nu \geq -r_i \qquad i = 1, \ldots, m.$$

The linear ℓ_∞ problem (9.2.23) leads to the problem

$$\begin{array}{ll} Minimize & \nu \\ \mathbf{x} \in \mathbb{R}^n,\ \nu \in \Re & \end{array} \tag{9.2.24}$$

$$Subject\ to \quad \begin{bmatrix} X & \mathbf{e} \\ -X & \mathbf{e} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \nu \end{bmatrix} \geq \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix} \tag{9.2.25}$$

$$\nu \geq 0 \tag{9.2.26}$$

This is a linear programming problem in $n+1$ variable $\mathbf{x}$ and ν, with $2m$ general linear inequality constraints. The inequality constraints (9.2.25) imply that $\nu \geq ||\mathbf{r}||_\infty$, while the non-negativity constraint (9.2.26) is redundant. Solving this linear programming problem will provide a solution to the linear ℓ_∞ approximation problem. Again the special structure of this linear programming problem can be exploited, or direct descent method developed, to produce more efficient ways of solving the linear ℓ_∞ problem.

9.2.4 Nonlinear Least Squares

If $\phi(\mathbf{x}; t)$ is not a linear function of the parameters $\mathbf{x}$ the the use of the 2-norm leads to the nonlinear least squares problem

$$\begin{array}{ll} Minimize & \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \\ \mathbf{x} \in \mathbb{R}^n & \end{array} \tag{9.2.27}$$

$$\tag{9.2.28}$$

where

$$r_i(\mathbf{x}) = \phi(\mathbf{x}; t_i) - y_i \qquad i = 1, \ldots, m.$$

Define the m by n Jacobian matrix

$$J_{ij}(\mathbf{x}) = \frac{\partial r_i(\mathbf{x})}{x_j} \qquad i = 1, \ldots, m \quad j = 1, \ldots, n.$$

This is a nonlinear optimization problem with gradient

$$\mathbf{g}(\mathbf{x}) = 2J^T(\mathbf{x})\mathbf{r}(\mathbf{x}) \tag{9.2.29}$$

and Hessian

$$G(x) = 2J^T(\mathbf{x})J(\mathbf{x}) + 2\sum_{i=1}^{m} r_i(\mathbf{x})\nabla^2 r_i(\mathbf{x}). \tag{9.2.30}$$

The Hessian consists of a term $J^T(\mathbf{x})J(\mathbf{x})$ which is non-negative definite (positive definite if J(x) is full rank) and a term which is zero if either

1. $\mathbf{r}(\mathbf{x}) = 0$, so $\mathbf{x}$ is a solution to the system of nonlinear equations $\phi(\mathbf{x}; t_i) = y_i$ for $i = 1, \ldots, m$.
2. $\nabla r_i(\mathbf{x}) = 0$ for $i = 1, \ldots, m$ (for example if $\mathbf{r}(\mathbf{x})$ is linear).

There are methods (for example the Gauss-Newton method) which are suitable for the *small residual* case when $G(\mathbf{x}) \approx 2J^T(\mathbf{x})J(\mathbf{x})$, while other methods are designed for the *large residual* case when the second term in (9.2.30) is significant.

9.2.5 Nonlinear ℓ_1

Introducing additional variables $\mathbf{u} \in \mathbb{R}^m$ corresponding to the positive part of $\mathbf{r}(\mathbf{x})$, and $\mathbf{v} \in \mathbb{R}^m$ corresponding to the negative part of $\mathbf{r}(\mathbf{x})$ leads to the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n,\, \mathbf{u}, \mathbf{v} \in \mathbb{R}^m}{Minimize} & \mathbf{e}^T\mathbf{u} + \mathbf{e}^T\mathbf{v} \qquad (9.2.31) \\ Subject\ to & \mathbf{u} - \mathbf{v} - \mathbf{r}(\mathbf{x}) = 0 \\ & \mathbf{u} \geq 0,\ \mathbf{v} \geq 0 \qquad (9.2.32) \\ & \mathbf{u}^T\mathbf{v} = 0. \qquad (9.2.33) \end{array}$$

This is a nonlinearly constrained optimization problem in $n + 2m$ variables.

9.2.6 Nonlinear ℓ_∞ and Minimax

Introducing an extra variable ν corresponding to $||\mathbf{r}(\mathbf{x})||_\infty$ leads to the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n,\, \nu \in \Re}{Minimize} & \nu \qquad (9.2.34) \\ Subject\ to & \nu\mathbf{e} + \mathbf{r}(\mathbf{x}) \geq 0 \\ & \nu\mathbf{e} - \mathbf{r}(\mathbf{x}) \geq 0 \\ & \nu \geq 0 \end{array}$$

The minimax problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & \max\limits_{i=1,\ldots,m} r_i(\mathbf{x}) \qquad (9.2.35) \\ & \qquad (9.2.36) \end{array}$$

leads to the nonlinear programming problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n,\, \nu \in \Re}{Minimize} & \nu \qquad (9.2.37) \\ Subject\ to & \nu\mathbf{e} - \mathbf{r}(\mathbf{x}) \geq 0 \end{array}$$

9.2.7 Exercises

1. Derive the solution (9.2.13) to the generalized least squares problem (9.2.12).
2. (a) For the positive definite quadratic function

$$q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{d}^T\mathbf{x} + q_0$$

find an expression for $q(\mathbf{x}^*)$ where $\mathbf{x}^*$ is the minimizer of $q(\mathbf{x})$.

(b) Show that the sum of squares error (9.2.8) for linear least squares can be calculated by

$$||\mathbf{r}^*||_2^2 = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T X\mathbf{x}^*. \qquad (9.2.38)$$

3. Consider the linearly constrained least squares problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} & \frac{1}{2}||\mathbf{x}||_2^2 \\ Subject\ to & A\mathbf{x} = \mathbf{b} \end{array}$$

where A is a t by n matrix, with $t < n$.

(a) Find the solution in terms of the generalized inverse of A.
Note: As $t < n$ the generalized inverse $A^+ = A^T(AA^T)^{-1}$ and $AA^+ = I_t$ but $A^+A \neq I_n$.

(b) Find the solution in terms of the QR factorization of A^T.

4. Consider approximating the data given in Table 9.2.1 by $\phi(x;t) = x_1 + x_2 t$.

t	1	2	3
$\mathbf{y}$	3	$5+\epsilon$	7

Table 9.2.1: Linear approximation data

(a) Find the least squares solution $\mathbf{x}^*(\epsilon)$ and $||\mathbf{r}(\mathbf{x}^*)||_2$ as functions of ϵ.

(b) Find the ℓ_1 minimizer $\mathbf{x}^*(\epsilon)$ and $||\mathbf{r}(\mathbf{x}^*)||_1$ for $\epsilon = 0, 1, 10, 100$.

(c) Find the ℓ_∞ minimizer $\mathbf{x}^*(\epsilon)$ and $||\mathbf{r}(\mathbf{x}^*)||_\infty$ for $\epsilon = 0, 1, 10, 100$.

(d) Plot the data and the best approximations using the $1, 2$ and ∞ norms for $\epsilon = 0, 1, 10, 100$, and comment on their behaviour.

5. Find the best least squares approximation to

$$\phi(\mathbf{x};t) = x_1 + x_2 e^{-x_4 t} + x_3 e^{-x_5 t}$$

to the data listed in Table 9.2.2 from the Research School of Chemistry, A.N.U. Use the nonlinear

i	t_i	y_i	i	t_i	y_i	i	t_i	y_i
1	0	0.844	12	110	0.718	23	220	0.478
2	10	0.908	13	120	0.685	24	230	0.467
3	20	0.932	14	130	0.658	25	240	0.457
4	30	0.936	15	140	0.628	26	250	0.448
5	40	0.925	16	150	0.603	27	260	0.438
6	50	0.908	17	160	0.580	28	270	0.431
7	60	0.881	18	170	0.558	29	280	0.424
8	70	0.850	19	180	0.538	30	290	0.420
9	80	0.818	20	190	0.522	31	300	0.414
10	90	0.784	21	200	0.506	32	310	0.411
11	100	0.751	22	210	0.490	33	320	0.406

Table 9.2.2: Sargeson data for exponential fitting

optimization routine to solve this problem

(a) from the standard starting point

$$\mathbf{x}^{(s)} = \begin{bmatrix} 0.5 & 1.5 & -1.0 & 0.01 & 0.02 \end{bmatrix}^T.$$

(b) from $10\mathbf{x}^{(s)}$.

(c) from $100\mathbf{x}^{(s)}$.

A problem that can occur with exponential functions is overflow (generating numbers that are too large to be represented on a computer) On this problem these difficulties (if they occur) can be solved by adding the simple bounds $-10 \leq x_j \leq 10$ for $j = 1, \ldots, 5$.

How much use is being made of the structure of the optimization problem?

Plot the best least squares approximation on $\Gamma = [0, 320]$.

6. Use a nonlinear optimization routine to minimize the sum of squares obtained from the Caelli data in Example 1.2.11,

   (a) from the standard starting point $\mathbf{x}^{(s)}$.

   (b) from $10\mathbf{x}^{(s)}$.

   (c) from $100\mathbf{x}^{(s)}$.

   Plot the best least squares approximation on $\Gamma = [0, 6.5]$.

   A difficulty that can occur with exponential functions is underflow (generating numbers that are so small that they are stored as zero on a computer). On this problem these difficulties (if they occur) can be solved by adding the simple bounds $0 \leq x_j \leq 10$ for $j = 1, \ldots, 11$.

9.2.8 References

1. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, (Dover Publications, 1972).

2. B. M. Averick, R. G. Carter and J. J. Moré, "The MINPACK-2 test problem collection", *Report ANL/MCS-TM-150*, Mathematics and Computer Science Division, Argonne National Laboratory, (May 1991).

3. B. M. Averick and J. J. Moré, "User Guide for the MINPACK-2 test problem collection", *Report ANL/MCS-TM-157*, Mathematics and Computer Science Division, Argonne National Laboratory, (October 1991).

4. E. W. Cheney, *Introduction to Approximation Theory*, (McGraw-Hill, 1966).

5. M. R. Osborne, *Finite Algorithms in Optimization and Data Analysis*, (John Wiley, 1985).

6. M. J. D. Powell, *Approximation Theory and Methods*, (Cambridge University Press, 1981).

7. G. A. Watson, *Approximation Theory and Numerical Methods*, (John Wiley, 1980).

Chapter 10

Nonsmooth Optimization [H]

10.1 Introduction

Nonsmooth optimization deals with the minimization (or maximization) of functions which are not continuously differentiable everywhere in the domain of interest. Much of the theory has developed from the theory of convex functions (see Chapter 2 and [Roc70]).

Nonsmooth optimization is based around generalizations of the idea of a gradient. For convex functions this is provided by the subdifferential, while for locally Lipschitz functions one possibility is the Clarke generalised gradient [Cla83].

A general approach to numerical methods nonsmooth optimization is *bundle methods* [HL93a, HL93b], where a single subgradient at each iterate is used to build an approximation to the generalized subdifferential. If it is only possible to calculate a single element of the generalized subdifferential at a point then bundle methods should be considered. Bundle methods are generally only linearly convergent.

However when the nonsmooth function is made up from a convex composition of smooth functions it is often possible to exploit the structure of the problem to obtain the full generalised gradient and to produce more efficient and robust numerical methods.

10.2 Convex optimization

10.2.1 Derivatives of convex functions

A convex function f defined on a convex set Ω may be extended to a convex function $\bar{f} : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ by

$$\bar{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega; \\ \infty & \text{otherwise.} \end{cases}$$

This can be a theoretically useful way of expressing the constrained optimization problem

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}).$$

However the fact that f may take the value $+\infty$ introduces additional complications, and means that it is not directly useful for numerical methods. Hence it is subsequently assumed that f is a convex function defined on $\mathbb{R}^n$.

A convex fnction $f : \mathbb{R}^n \to \mathbb{R}$ is continuous, but not necessarily continuously differentialbe everywhere. Functions which are not continuously differentiable at all points of their domain are called *nonsmooth*. The classic example on $\mathbb{R}$ is

$$f(x) = |x| = \max\{x, -x\} \tag{10.2.1}$$

which is not continuously differentiable at $x = 0$. Another example is the polyhedral convex function (2.2.8) which is not continuously differentiable at points where more than one component functin achieves the maximum. An example of a convex functions which is continuously differentiable, but not twice

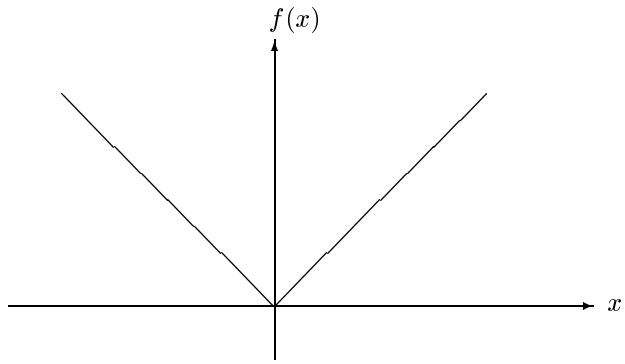


Figure 10.2.1: $f(x) = |x| = \max\{x, -x\}$

continuously at all points, is the M-etimation function (2.2.9). Examples of convex functions that have derivatives of all orders are the positive definite quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{g}_0^T\mathbf{x} + f_0$ and the one dimensional function $f(x) = e^{-x}$.

10.2.2 Directional Derivatives

The definition (2.2.1) of a convex function implies that the chord joining $f(\mathbf{x})$ and $f(\mathbf{y})$ always overestimates the function. On the other hand the tangent line (if it exists) always underestimates the function (see Figure 10.2.2). At points where f is nonsmooth the idea of the slope of the function in a given direction is required. For a convex function the one-sided *directional derivative*

$$f'(\mathbf{x}; \mathbf{d}) = \lim_{t \downarrow 0} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t} \tag{10.2.2}$$

is defined at any point $\mathbf{x}$ and for any direction $\mathbf{d} \in \mathbb{R}^n$. If f is differentiable at $\mathbf{x}$ then

$$f'(\mathbf{x}; \mathbf{d}) = \mathbf{d}^T \nabla f(\mathbf{x}).$$

An essential feature of the directional derivative is its one-sided nature as the definition (10.2.2) only involves $t > 0$. The length of the direction vector $\mathbf{d}$ is not important as the directional derivative $f'(\mathbf{x}; \mathbf{d})$ is *positively homogeneous* as a function of $\mathbf{d}$. That is, for any $\mathbf{x}, \mathbf{d} \in \mathbb{R}^n$

$$f'(\mathbf{x}; \alpha\mathbf{d}) = \alpha f'(\mathbf{x}; \mathbf{d}) \qquad \forall \alpha \geq 0. \tag{10.2.3}$$

Example 10.2.1 *The function*

$$f(x) = \max\{\, -x, x^2 \,\} \tag{10.2.4}$$

is sketched in Figure 10.2.2.

It is easily verified by direct computation that

$$f'(0; +1) = 0 \;\; \textit{and} \;\; f'(0; -1) = 1,$$

or equivalently that

$$f'(0; d) = \begin{cases} 0 & d > 0; \\ -d & d < 0. \end{cases} \tag{10.2.5}$$

10.2.3 Subgradients and Subdifferentials

The appropriate generalization of the concept of the gradient of a smooth function to convex functions is provided by the subgradient and subdifferential. A *subgradient* of f at a point $\mathbf{x}$ is a vector $\mathbf{u} \in \mathbb{R}^n$ such that

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{u}^T(\mathbf{y} - \mathbf{x}) \qquad \forall \mathbf{y} \in \mathbb{R}^n. \tag{10.2.6}$$

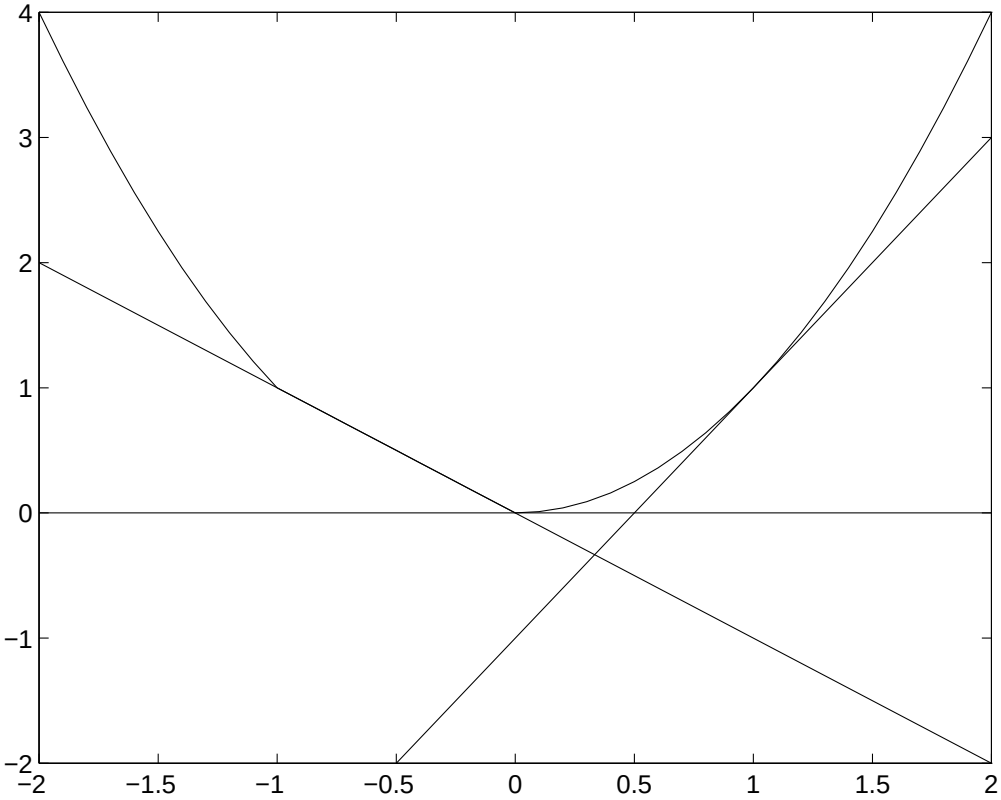


Figure 10.2.2: Subgradients of $max\{ -x, x^2 \}$

The set of all subgradients is the *subdifferential*

$$\partial f(\mathbf{x}) = \{ \mathbf{u} \in \mathbb{R}^n : f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{u}^T(\mathbf{y} - \mathbf{x}) \quad \forall \mathbf{y} \in \mathbb{R}^n \}. \tag{10.2.7}$$

Proposition 10.2.2 *If $f : \mathbb{R}^n \to \mathbb{R}$ is convex then the subdifferential $\partial f(\mathbf{x})$ has the following properties:*

1. *The subdifferential $\partial f(\mathbf{x})$ is a non-empty compact (closed and bounded), convex set in $\mathbb{R}^n$.*
2. *The directional derivative is the maximum of the inner product of subgradients and the direction:*

$$f'(\mathbf{x}; \mathbf{d}) = \max_{\mathbf{u} \in \partial f(\mathbf{x})} \mathbf{u}^T \mathbf{d} \qquad \forall \mathbf{d} \in \mathbb{R}^n. \tag{10.2.8}$$

3. *Conversely the subdifferential is the set of all vectors for which the inner product with the direction d is less than or equal to the directional derivative:*

$$\partial f(\mathbf{x}) = \{ \mathbf{u} \in \mathbb{R}^n : f'(\mathbf{x}; \mathbf{d}) \geq \mathbf{u}^T \mathbf{d} \quad \forall \mathbf{d} \in \mathbb{R}^n \}. \tag{10.2.9}$$

4. *The directional derivative underestimates the function, that is*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + f'(\mathbf{x}; \mathbf{y} - \mathbf{x}) \qquad \forall \mathbf{y} \tag{10.2.10}$$

5. *f is differentiable at $x \iff \partial f(\mathbf{x}) = \{ \nabla f(\mathbf{x}) \}$.*
6. *The subdifferential $\partial f(\mathbf{x})$ is the convex hull of all the limits of gradients $\nabla f(\mathbf{y})$ at points $\mathbf{y}$ where f is differentiable and which converge to $\mathbf{x}$, that is*

$$\partial f(\mathbf{x}) = \operatorname{conv} \{ \lim \nabla f(\mathbf{x}) : \mathbf{y} \to \mathbf{x}, \nabla f(\mathbf{y}) \text{ exists} \}.$$

7. *The subdifferential is* upper-semicontinuous, *that is*

$$\mathbf{y}^{(k)} \to \mathbf{x}, \quad \mathbf{u}^{(k)} \in \partial f(\mathbf{y}^{(k)}), \quad \mathbf{u}^{(k)} \to \mathbf{u} \Rightarrow \mathbf{u} \in \partial f(\mathbf{x}).$$

The function (10.2.4) in Example 10.2.1 is differentiable for $x \neq 0$ and $x \neq 1$ and

$$\nabla f(x) = \begin{cases} 2x & x > 0 \text{ or } x < -1; \\ -1 & -1 < x < 0. \end{cases}$$

Thus

$$\begin{aligned} \partial f(0) &= \operatorname{conv}\{\,0, -1\,\} = [-1, 0] \\ \partial f(-1) &= \operatorname{conv}\{\,-1, -2\,\} = [-2, -1] \end{aligned}$$

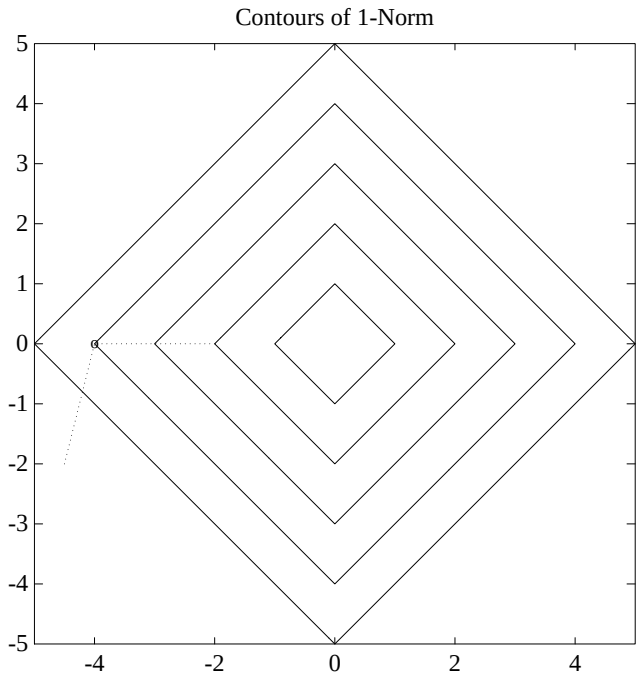


Figure 10.2.3: Contours of $||\mathbf{x}||_1$

Example 10.2.3 *For $n = 2$ the contours of $f(\mathbf{x}) = ||\mathbf{x}||_1 = |x_1| + |x_2|$ are sketched in Figure 10.2.3. Consider a point $\overline{\mathbf{x}} = [\,-\eta \quad 0\,]^T$ where $\eta > 0$. Then as f is piecewise linear*

$$\begin{aligned} \partial f(\overline{\mathbf{x}}) &= \operatorname{conv}\{\,[\,-1 \quad -1\,]^T,\ [\,-1 \quad 1\,]^T\,\} \\ &= \{\,\mathbf{u} \in \mathbb{R}^2 : u_1 = -1,\ -1 \leq u_2 \leq 1\,\}. \end{aligned} \tag{10.2.11}$$

Hence from (10.2.8)

$$f'(\overline{\mathbf{x}}; \mathbf{d}) = -d_1 + \max_{|u_2| \leq 1} u_2 d_2 = -d_1 + |d_2|,$$

so $\bar{\mathbf{d}} = [\,2 \quad 0\,]^T$ gives $f'(\overline{\mathbf{x}}; \bar{\mathbf{d}}) = -2$, while $\hat{\mathbf{d}} = [\,-0.5 \quad -2\,]^T$ gives $f'(\overline{\mathbf{x}}; \hat{\mathbf{d}}) = 2.5$ (see Figure 10.2.3).

In many practical applications the functions f is a polyhedral convex function

$$f(\mathbf{x}) = \max_{i=1,\ldots,m} \mathbf{a}_i^T \mathbf{x} + \beta_i.$$

Let x^+ denote the vector in $\mathbb{R}^n$ with components $x_i^+ = \max\{x_i, 0\}$. The following functions are polyhedral convex functions where $\beta_i \equiv 0$ for $i = 1, \ldots, m$ and A is the n by m matrix with columns $\mathbf{a}_i \in \mathbb{R}^n$ $i = 1, \ldots, m$ specified in (10.2.12a) to (10.2.12e).

$$\max_{i=1,\ldots,n} x_i \quad : \quad m = n; \quad A = I \tag{10.2.12a}$$

$$||\mathbf{x}||_\infty \quad : \quad m = 2n; \quad A = \begin{bmatrix} I & -I \end{bmatrix} \tag{10.2.12b}$$

$$||\mathbf{x}^+||_\infty \quad : \quad m = n+1; \quad A = \begin{bmatrix} I & 0 \end{bmatrix} \tag{10.2.12c}$$

$$||\mathbf{x}||_1 \quad : \quad m = 2^n; \quad \mathbf{a}_i \text{ are all possible combinations of } \pm 1 \tag{10.2.12d}$$

$$||\mathbf{x}^+||_1 \quad : \quad m = 2^n; \quad \mathbf{a}_i \text{ are all possible combinations of } 0, 1 \tag{10.2.12e}$$

For a polyhedral convex function only the component functions $\mathbf{a}_i^T \mathbf{x} + \beta_i$ which achieve the maximum are important in a neighbourhood of $\mathbf{x}$. Define the set of active component functions $\mathcal{A}(\mathbf{x})$ by

$$\mathcal{A}(\mathbf{x}) = \{\, i \in 1, \ldots, m : \mathbf{a}_i^T \mathbf{x} + \beta_i = f(\mathbf{x}) \,\}. \tag{10.2.13}$$

Then

$$\begin{aligned} \partial f(\mathbf{x}) &= \operatorname{conv}\{\, \mathbf{a}_i : i \in \mathcal{A}(\mathbf{x}) \,\} \\ &= \left\{ \mathbf{u} \in \mathbb{R}^n : \mathbf{u} = \sum_{i \in \mathcal{A}(y)} \alpha_i \mathbf{a}_i, \quad \sum_{i \in \mathcal{A}(\mathbf{x})} \alpha_i = 1, \quad \alpha_i \geq 0 \; i \in \mathcal{A}(\mathbf{x}) \right\}. \end{aligned} \tag{10.2.14}$$

The characterization (10.2.14) of the subdifferential of a polyhedral convex function can be used to calculate the subdifferentials of the functions (10.2.12a) to (10.2.12e). Let $\mathbf{e} = [\, 1 \quad \cdots \quad 1 \,]^T \in \mathbb{R}^n$ and $|\mathbf{x}| = [\, |x_1| \quad \cdots \quad |x_n| \,]^T$, so

$$\mathbf{e}^T \mathbf{x} = \sum_{i=1}^n x_i \quad \text{and} \quad \mathbf{e}^T |\mathbf{x}| = \sum_{i=1}^n |x_i| = ||\mathbf{x}||_1.$$

This leads to the following characterizations

$$\partial \max_{i=1,\ldots,m} x_i = \left\{ \mathbf{u} \in \mathbb{R}^m : \begin{array}{l} \mathbf{e}^T \mathbf{u} = 1, \mathbf{u} \geq 0 \\ x_i < \max_i x_i \Rightarrow u_i = 0 \end{array} \right\}. \tag{10.2.15}$$

For the functions (10.2.12b) and (10.2.12c) the only difficulty is taking care of the case when $f(\mathbf{x}) = 0$. If $\mathbf{x} = 0$ then

$$\partial ||\mathbf{x}||_\infty = \{\, \mathbf{u} \in \mathbb{R}^n : ||u||_1 \leq 1 \,\} \tag{10.2.16}$$

while for $\mathbf{x} \neq 0$

$$\partial ||\mathbf{x}||_\infty = \left\{ \mathbf{u} \in \mathbb{R}^n : ||u||_1 = 1, \begin{array}{l} |x_i| < ||\mathbf{x}||_\infty \Rightarrow u_i = 0, \\ |x_i| = ||\mathbf{x}||_\infty \Rightarrow u_i x_i \geq 0 \end{array} \right\}. \tag{10.2.17}$$

Similarly

$$\partial ||\mathbf{x}^+||_\infty = \left\{ \mathbf{u} \in \mathbb{R}^n : \begin{array}{l} \mathbf{e}^T \mathbf{u} \leq 1, \mathbf{u} \geq 0, \\ x_i^+ < ||\mathbf{x}^+||_\infty \Rightarrow u_i = 0, \\ \mathbf{x}^+ \neq 0 \Rightarrow \mathbf{e}^T \mathbf{u} = 1 \end{array} \right\}. \tag{10.2.18}$$

However, in the case of the functions involving the 1-norm, a lot of the structure can be lost by treating the function as a general polyhedral convex function. A more productive approach is to identify what is causing the nonsmooth nature of f. For instance with the max function (10.2.12a) or the polyhedral convex function (2.2.8) f is nonsmooth when more than 1 element achieves the maximum, so $|\mathcal{A}(\mathbf{x})| > 1$.

However for the 1-norm (10.2.12d) the function f is nonsmooth when one or more of the variables $x_i = 0$. The following characterizations of the subdifferentials may then be derived.

$$\partial||\mathbf{x}||_1 = \left\{ \mathbf{u} \in \mathbb{R}^n : \begin{array}{ll} |u_i| \leq 1 & \text{if } x_i = 0; \\ u_i = sign(x_i) & \text{if } x_i \neq 0. \end{array} \right\} \tag{10.2.19}$$

$$\partial||\mathbf{x}^+||_1 = \left\{ \mathbf{u} \in \mathbb{R}^m : \begin{array}{ll} u_i = 1 & \text{if } x_i > 0; \\ 0 \leq u_i \leq 1 & \text{if } x_i = 0; \\ u_i = 0 & \text{if } x_i < 0. \end{array} \right\} \tag{10.2.20}$$

The 2-norm $||\mathbf{x}||_2 = \sqrt{\mathbf{x}^T\mathbf{x}}$ is not a polyhedral convex function, but is smooth except when $\mathbf{x} = 0$ (typically this is overcome by minimizing $||\mathbf{x}||_2^2$ rather than $||\mathbf{x}||_2$). The subdifferential is

$$\partial||\mathbf{x}||_2 = \left\{ \begin{array}{ll} \{ \mathbf{u} \in \mathbb{R}^n : \mathbf{u} = \frac{\mathbf{x}}{||\mathbf{x}||_2} \} & \text{if } \mathbf{x} \neq 0; \\ \{ \mathbf{u} \in \mathbb{R}^n : ||\mathbf{u}||_2 \leq 1 \} & \text{if } \mathbf{x} = 0. \end{array} \right. \tag{10.2.21}$$

For any polyhedral convex function the problem of minimizing $f(\mathbf{x})$ is equivalent to the linear programming problem

$$\begin{array}{ll} \textit{Minimize} & \nu \\ \mathbf{x} \in \mathbb{R}^m,\ \nu \in \mathbb{R} & \\ \textit{Subject to} & \nu \geq \mathbf{a}_i^T\mathbf{a} + \beta_i \quad i = 1, \ldots, m \end{array}$$

However more efficient methods can be developed by fully utilizing the structure of the particular polyhedral convex function.

10.2.4 Optimality Conditions

If f is convex on $\mathbb{R}^n$ then any local minimizer of f on $\mathbb{R}^n$ is a global minimizer of f. The convexity of f also allows one to establish necessary and sufficient optimality conditions. Thus

$$\mathbf{x}^* \text{ is a minimizer of } f \iff f(\mathbf{x}) \geq f(\mathbf{x}^*) \quad \forall\ \mathbf{x}. \tag{10.2.22}$$

and

$$\mathbf{x}^* \text{ is a strict minimizer of } f \iff f(\mathbf{x}) > f(\mathbf{x}^*) \quad \forall\ \mathbf{x} \neq \mathbf{x}^*. \tag{10.2.23}$$

From the definition (10.2.2) of the directional derivative and (10.2.22) is follows that

$$\mathbf{x}^* \text{ is a minimizer of } f \iff f'(\mathbf{x}^*; \mathbf{d}) \geq 0 \quad \forall\ \mathbf{d} \in \mathbb{R}^n, \tag{10.2.24}$$

and that

$$f'(\mathbf{x}^*; \mathbf{d}) > 0 \quad \forall\ \mathbf{d} \neq 0 \Rightarrow \mathbf{x}^* \text{ is a strict minimizer of } f. \tag{10.2.25}$$

The function (10.2.4) shows that the converse of (10.2.25) is not true.

Theorem 10.2.4 (First order conditions) *A point $\mathbf{x}^*$ is a minimizer of a convex function f on $\mathbb{R}^n$ if and only if $0 \in \partial f(\mathbf{x}^*)$.*

Proof As $\mathbf{x}^*$ is a minimizer of f if and only if $f'(\mathbf{x}^*; \mathbf{d}) \geq 0\ \forall \mathbf{d} \in \mathbb{R}^n$ the result is established by proving that

$$0 \in \partial f(\mathbf{x}^*) \iff f'(\mathbf{x}^*; \mathbf{d}) \geq 0 \quad \forall \mathbf{d} \in \mathbb{R}^n. \tag{10.2.26}$$

Firstly if $0 \in \partial f(\mathbf{x}^*)$ then (10.2.8) immediately implies that $f'(\mathbf{x}^*; \mathbf{d}) \geq 0$. Conversely suppose that $f'(\mathbf{x}^*; \mathbf{d}) \geq 0$ for all $\mathbf{d} \in \mathbb{R}^n$ but that $0 \notin \partial f(\mathbf{x}^*)$. As $\partial f(\mathbf{x})$ is a compact convex set the Separating Hyperplane Theorem 2.1.11 implies that there exists a $\mathbf{v} \in \mathbb{R}^n$ such that

$$0 > \mathbf{v}^T\mathbf{u} \qquad \forall \mathbf{u} \in \partial fh(\mathbf{x}^*).$$

Equation (10.2.9) implies that

$$f'(\mathbf{x}^*; v) = \max_{\mathbf{u} \in \partial f(\mathbf{x}^*)} \mathbf{v}^T\mathbf{u} < 0$$

which is a contradiction. Hence $0 \in \partial f(\mathbf{x}^*)$. ∎

Theorem 10.2.5 *If* $0 \in \text{int}\partial f(\mathbf{x}^*)$ *then* $\mathbf{x}^*$ *is a strict minimizer of* f *on* $\mathbb{R}^n$.

Proof This is established by showing that

$$0 \in \text{int}\partial f(\mathbf{x}^*) \iff f'(\mathbf{x}^*; \mathbf{d}) > 0 \quad \forall \mathbf{d} \neq 0. \tag{10.2.27}$$

∎

Example 10.2.6 *Consider the function* $f(\mathbf{x}) = |x_1|$ *on* $\mathbb{R}^2$. *At the point* $\overline{\mathbf{x}} = 0$

$$\partial f(\overline{\mathbf{x}}) = \{\, \mathbf{u} \in \mathbb{R}^2 : -1 \leq u_1 \leq 1,\ u_2 = 0 \,\}$$

so $0 \in \partial f(\overline{\mathbf{x}})$ *and* $\overline{\mathbf{x}}$ *is a minimizer of* f. *However* $0 \notin \text{int}\partial f(\overline{\mathbf{x}})$ *as the interior of* $\partial f(\overline{\mathbf{x}})$ *is empty. Hence, from* $\partial f(\overline{\mathbf{x}})$ *it is not possible to determine if* $\overline{\mathbf{x}}$ *is a strict local minimizer of* f. *For this example* $\overline{\mathbf{x}}$ *is not a strict minimizer, as any point* $x = [\,0 \quad x_2\,]^T$ *is a minimizer of* f. *On the other hand* $\hat{f}(\mathbf{x}) = |x_1| + x_2^2$ *has the same subdifferential at* $\overline{\mathbf{x}}$ *and* $\overline{\mathbf{x}}$ *is a strict minimizer of* $\hat{f}$.

10.2.5 Exercises

1. Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. Prove the following results:

   (a) Property (10.2.10)
   $$f(\mathbf{y}) \geq f(\mathbf{x}) + f'(\mathbf{x}; \mathbf{y} - \mathbf{x}) \qquad \forall \mathbf{y}$$

   (b) Equation (10.2.27)
   $$0 \in \text{int}\partial f(\mathbf{x}^*) \iff f'(\mathbf{x}^*; \mathbf{d}) > 0 \quad \forall \mathbf{d} \neq 0.$$

10.3 Convex Composite Optimization

10.3.1 Convex composite functions

Many optimization problems, including data fitting problems, can be expressed as

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{Minimize}} \quad \phi(\mathbf{x})$$

where

$$\phi(\mathbf{x}) = h(\mathbf{r}(\mathbf{x})), \tag{10.3.1}$$

$\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$ is smooth (at least twice continuously differentiable), and $h : \mathbb{R}^m \to \mathbb{R}$ is convex but not necessarily differentiable. The function (10.3.1) is a *convex composite function*. Convex composite functions also include the penalty functions

$$\phi(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{c}(\mathbf{x}))$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$ are smooth and $h : \mathbb{R}^m \to \mathbb{R}$ is convex. A convex composite function is not necessarily convex, for example $\phi(x) = |x^2 - 1|$ for $x \in \mathbb{R}$, where $h(c) = |c|$ is convex and $c(x) = x^2 - 1$ is smooth.

Typical functions h are

$$h(\mathbf{y}) = \max_{i=1,\ldots,m} y_i \tag{10.3.2}$$
$$h(\mathbf{y}) = ||\mathbf{y}||_\infty \tag{10.3.3}$$
$$h(\mathbf{y}) = ||\mathbf{y}^-||_\infty \tag{10.3.4}$$
$$h(\mathbf{y}) = ||\mathbf{y}||_1 \tag{10.3.5}$$
$$h(\mathbf{y}) = ||\mathbf{y}^-||_1 \tag{10.3.6}$$
$$h(\mathbf{y}) = ||\mathbf{y}||_2 \tag{10.3.7}$$

where

$$y_i^- = \max\{\,0, -y_i\,\} \qquad i = 1, \ldots, m.$$

All the functions (10.3.2) to (10.3.7) are convex functions of y. The theory of convex functions (see Chapter 2) can be used to provide generalizations of the idea of a gradient, optimality conditions and algorithms for convex composite functions. In many cases there is a close relationship between constrained nonlinear programming problems where all the functions are smooth, and unconstrained convex composite optimization where the function may be nonsmooth.

Convex composite optimization has developed from the work of Rockafellar [Roc70] on convex analysis, and the extesnion of this to locally Lipschitz functions by Clarke [Cla83]. Polak [Pol87] surveys some of the foundations of nondifferentiable optimization. Special cases, like the minimax problem [DM74] and data fitting problems [Osb85], have been treated separately. The theory and alogrithms motivated by engineering problems are coved extensivly in Polak /citeNSO:Pol97. When the nonsmooth objective function has less structure then bundle methods [Kiw85, HL93a, HL93b] are appropriate. This chapter follows the work of Fletcher [Fle87].

10.3.2 Generalized gradients

There are many ways of extending the ideas of directional derivatives and subdifferentials to certain classes of nonsmooth nonconvex functions. For example Clarke (1983) has pioneered the development of the generalized gradient of locally Lipschitz functions. A function $\phi(\mathbf{x})$ is *locally Lipschitz* if for any $\mathbf{x}$ there exists a neighbourhood $N(\mathbf{x})$ and a constant L such that

$$|\phi(\mathbf{x}) - \phi(\mathbf{y})| \le L||\mathbf{x} - \mathbf{y}|| \quad \forall \mathbf{y} \in N(\mathbf{x}).$$

Locally Lipschitz functions include all convex composite functions, but are too general a class of functions in many cases (for example a concave function is locally Lipschitz, but cannot be represented as a convex composite function).

Following Fletcher (1987) the concepts of a directional derivative and subdifferential can be directly extended from convex functions to convex composite functions. As $c(\mathbf{x})$ and $f(\mathbf{x})$ are smooth then

$$\begin{aligned} \mathbf{c}(\mathbf{x} + \alpha\mathbf{d}) &= \mathbf{c}(\mathbf{x}) + \alpha A(\mathbf{x})^T\mathbf{d} + o(\alpha) \\ f(\mathbf{x} + \alpha\mathbf{d}) &= f(\mathbf{x}) + \alpha\nabla f(\mathbf{x})^T\mathbf{d} + o(\alpha) \end{aligned}$$

where $A(\mathbf{x}) = [\nabla c_1(\mathbf{x}) \ \ldots \ \nabla c_m(\mathbf{x})]$ is the matrix whose columns are the gradients of the functions $c_i(\mathbf{x})$. (The Jacobian of $\mathbf{c}(\mathbf{x})$ is usually $A(\mathbf{x})^T$). Using these expansions it can be established that

$$\phi'(\mathbf{x}; \mathbf{d}) = \nabla f(\mathbf{x})^T\mathbf{d} + h'(\mathbf{c}(\mathbf{x}); A(\mathbf{x})^T\mathbf{d}).$$

Then using (10.2.8) it follows that

$$\phi'(\mathbf{x}; \mathbf{d}) = \max_{\lambda \in \partial h(\mathbf{c})} \mathbf{d}^T(\nabla f(\mathbf{x}) + A(\mathbf{x})\lambda).$$

From (10.2.9) this suggests using

$$\partial\phi(\mathbf{x}) = \{\,\nabla f(\mathbf{x}) + A(\mathbf{x})\lambda : \lambda \in \partial h(\mathbf{c})\,\}. \tag{10.3.8}$$

Here λ is in the subdifferential of h with respect to the variables $\mathbf{c}$ evaluated at the point $\mathbf{c}(\mathbf{x})$. The set (10.3.8) is referred to as the *generalized gradient* or *generalized subdifferential*. It can be shown that (10.3.8) is a non-empty compact convex set. Moreover $\partial\phi(\mathbf{x})$ is a singleton if and only if ϕ is differentiable at $\mathbf{x}$. If ϕ is convex then the generalized subdifferential is the same as the subdifferential. In fact all the properties, except (10.2.10), in Proposition 10.2.2 are valid if h is replaced by ϕ.

10.3.3 Optimality conditions

A necessary condition for $\mathbf{x}^*$ to be a local minimizer of $\phi(\mathbf{x})$ is that $\phi'(\mathbf{x}^*; \mathbf{d}) \ge 0 \ \forall \mathbf{d}$. As in the convex case it can be established that

$$\phi'(\mathbf{x}^*; \mathbf{d}) \ge 0 \quad \forall \mathbf{d} \in \mathbb{R}^n \iff 0 \in \partial\phi(\mathbf{x}^*).$$

Theorem 10.3.1 (First order necessary condition) *If $\mathbf{x}^*$ is a local minimizer of $\phi(\mathbf{x})$ then there exists $\lambda^* \in \partial h(\mathbf{c}^*)$ such that*

$$\nabla f^* + A^*\lambda^* = \mathbf{0}. \tag{10.3.9}$$

A point $\overline{\mathbf{x}}$ with $0 \in \partial\phi(\overline{\mathbf{x}})$ is referred to as a *stationary point.*

It is convenient to write the optimality conditions in terms of the Lagrangian function

$$\mathcal{L}(\mathbf{x};\lambda) = f(\mathbf{x}) + \mathbf{c}(\mathbf{x})^T\lambda. \tag{10.3.10}$$

The necessary conditions are that there exist a $\lambda^* \in \partial h(\mathbf{c}^*)$ such that

$$\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*;\lambda^*) = \mathbf{0}.$$

This is similar to the necessary conditions for constrained optimization. In nonlinear programming the only restrictions on the Lagrange multipliers are non-negativity conditions for multipliers corresponding to constrains $c_i(\mathbf{x}) \geq 0$. In convex composite optimization the multipliers must lie in $\partial h(\mathbf{c}^*)$. Unlike nonlinear programming a constraint qualification is not required for the first order necessary conditions for convex composite functions.

A sufficient condition for $\mathbf{x}^*$ to be a strict local minimizer of ϕ is that $\phi'(\mathbf{x}^*;\mathbf{d}) > 0$ for all $\mathbf{d} \neq 0$. This is equivalent to the condition that

$$\mathbf{0} \in \operatorname{int}\partial\phi(\mathbf{x}^*).$$

As in the convex case the zero vector must be in the interior of $\partial\phi(\mathbf{x}^*)$ as a set in $\mathbb{R}^n$, not the relative interior of $\partial\phi(\mathbf{x}^*)$. This is analogous to the case in constrained smooth optimization when the first order conditions can be sufficient to determine a local minimizer at a vertex of the feasible region.

As in nonlinear programming second order information can be used to say more about the nature of a stationary point. If $\phi'(\mathbf{x};\mathbf{d}) > 0$ then $\mathbf{x}$ is a local minimizer of $\phi(\mathbf{x})$ along the ray in the direction $\mathbf{d}$ from $\mathbf{x}$. Thus the curvature of ϕ is only important in determining a minimum in directions $\mathbf{d}$ with zero directional derivative. Define

$$\mathcal{D}(\mathbf{x}) = \{\, \mathbf{d} \in \mathbb{R}^n : \phi'(\mathbf{x};\mathbf{d}) = 0 \,\}. \tag{10.3.11}$$

Theorem 10.3.2 (Second order necessary conditions) *If $\mathbf{x}^*$ is a local minimizer of ϕ then there exists $\lambda^* \in \partial h(\mathbf{c}^*)$ such that*

$$\nabla f^* + A^*\lambda^* = 0$$

and

$$\mathbf{d}^T\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*;\lambda^*)\mathbf{d} \geq 0 \qquad \forall \mathbf{d} \in \mathcal{D}(\mathbf{x}^*).$$

Theorem 10.3.3 (Second order sufficient conditions) *If there exists a $\lambda^* \in \partial h(\mathbf{c}^*)$ such that*

$$\nabla f^* + A^*\lambda^* = \mathbf{0}$$

and

$$\mathbf{d}^T\nabla^2_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*;\lambda^*)\mathbf{d} > 0 \qquad \forall \mathbf{d} \neq 0, \quad \mathbf{d} \in \mathcal{D}(\mathbf{x}^*)$$

then $\mathbf{x}^*$ is a strict local minimizer of $\phi(\mathbf{x})$.

A difficulty with both the second order necessary and sufficient conditions is the calculation of the set $\mathcal{D}(\mathbf{x}^*)$. The positive homogeneity of the directional derivative (10.2.3) implies that $\mathcal{D}(\mathbf{x}^*)$ is a cone in $\mathbb{R}^n$. However when h is a polyhedral convex function a regularity assumption can be used to show that $\mathcal{D}(\mathbf{x}^*)$ is a subspace of $\mathbb{R}^n$ and to obtain better formulae for calculating it (see Fletcher (1987) Lemmas 14.2.6 and 14.2.7).

Example 10.3.4 *Example 10.2.1 can be written as $\phi(\mathbf{x}) = h(\mathbf{c}(\mathbf{x}))$ where*

$$h(\mathbf{c}) = \max\{\, c_1, c_2 \,\} \quad \textit{and} \quad \mathbf{c}(\mathbf{x}) = \begin{bmatrix} -x \\ x^2 \end{bmatrix},$$

with $m = 2$ and $n = 1$. Let $\mathbf{x}^* = 0$. Then $\mathbf{c}^* = 0$, $\mathcal{A}^* = \{1, 2\}$ and from (10.2.15)

$$\partial h^* = \left\{ u \in \mathbb{R}^2 : \lambda = \begin{bmatrix} \lambda_1 \\ 1 - \lambda_1 \end{bmatrix} \quad \lambda_1 \in [0, 1] \right\}.$$

As $A(\mathbf{x}) = [-1 \ 2x]$ it follows that

$$\partial \phi^* = \{ u \in \mathbb{R} : u = A^* \lambda \quad \lambda \in \partial h^* \} = [-1, 0].$$

Hence $0 \in \partial \phi^*$ and $\nabla f^* + A^* \lambda^* = 0$ where $\lambda^* = [0 \ 1]^T \in \partial h(\mathbf{c}^*)$. Note that $\mathbf{0} \notin \text{int} \partial \phi^* = (-1, 0)$. From (10.2.5)

$$\mathcal{D}^* = \{ \mathbf{d} \in \mathbb{R} : d \geq 0 \}.$$

The Lagrangian function is

$$\mathcal{L}(\mathbf{x}; \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{c}(\mathbf{x}) = -\lambda_1 x + \lambda_2 x^2,$$

where $\lambda \in \partial h(\mathbf{c})$ and $c \equiv \mathbf{c}(\mathbf{x})$. As $f \equiv 0$ for this example

$$\nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \lambda) = \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(\mathbf{x}) = 2\lambda_2.$$

The Hessain of the Lagrangain function is evaluated at the point $\mathbf{x}^*$ and the multipliers $\lambda^* \in \partial h^*$ that satisfy $\nabla f^* + A^* \lambda^* = \mathbf{0}$. As $\lambda^* = [0 \ 1]^T$

$$\mathbf{d}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*; \lambda^*) \mathbf{d} = 2d^2$$

which is positive for any nonzero $\mathbf{d} \in \mathcal{D}^*$ (in this particular example the set $\mathcal{D}^*$ is irrelevant – why?). Hence both the second order necessary and the second order sufficient conditions are satisfied.

10.3.4 Numerical Methods

In the unconstrained minimization of a smooth function

$$\underset{\mathbf{x} \in \mathbb{R}^n}{Minimize} \qquad f(\mathbf{x})$$

a basic method is the *steepest descent method* where the search direction at a point $\mathbf{x}^{(k)}$ is $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$. A line search is then used to calculate

$$\alpha^{(k)} = \arg\min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$$

and the next iterate is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)}.$$

One attractive feature of the steepest method is that it is globally convergent to a stationary point (that is it converges to a stationary point from any starting point). More formally if $\nabla f(x)$ exists and is uniformly continuous on the level set $\{ \mathbf{x} : f(\mathbf{x}) < f(\mathbf{x}^{(1)}) \}$ and an (approximate) line search is used then any descent method for which ${\mathbf{d}^{(k)}}^T \nabla f(\mathbf{x}^{(k)}) \leq -\eta$ for some $\eta > 0$ independent of k then either $\nabla f(\mathbf{x}^{(k)}) = 0$ for some k, or $f(\mathbf{x}^{(k)}) \to -\infty$, or $\nabla f(\mathbf{x}^{(k)}) \to \mathbf{0}$. See Fletcher (1987) for more details. However the disadvantage of the steepest descent method is that it can have an arbitrarily slow rate of convergence even on a strictly convex quadratic function.

In nonsmooth optimization the steepest descent direction is defined by

$$\min_{||d|| \leq 1} h'(y; d). \tag{10.3.12}$$

When $0 \notin \partial h(y)$ the positive homogeneity of the directional derivative means that attention can be restricted to directions d with $||d|| = 1$. Then

$$\begin{aligned}
\min_{||d||=1} h'(y;d) &= \min_{||d||=1} \max_{u \in \partial h(y)} u^T d \\
&= \max_{u \in \partial h(y)} \min_{||d||=1} u^T d \\
&= \max_{u \in \partial h(y)} \frac{-u^T u}{||u||} \\
&= -||\bar{u}|| \qquad \text{where } \bar{u} = \arg \min_{u \in \partial h(y)} ||u||.
\end{aligned}$$

The step of swapping the min and the max requires justification, but is legitimate in this case. The steepest descent direction is determined by the element of the subdifferential closest to the origin. Let

$$Nr\ \partial h(y) \equiv \arg \min_{u \in \partial h(y)} ||u||, \tag{10.3.13}$$

so the steepest descent direction is $-\bar{u}$ where $\bar{u} = Nr\ \partial h(y)$.

Example 10.3.5 *Consider the function $h(y) = ||y||_1$ from Example 10.2.3. Let $\bar{y} = (-\eta\ 0)^T$ where $\eta > 0$. Then from (10.2.11)*

$$Nr\ \partial h(\bar{y}) = \arg \min_{|u_2| \le 1} \sqrt{(-1)^2 + u_2^2} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}.$$

Thus the steepest descent direction at $\bar{y}$ is

$$\bar{d} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

which is illustrated in Figure 10.2.3.

In contrast to the smooth case, the steepest descent method for nonsmooth optimization (even with exact line searches on a strictly convex function) may fail to converge to a stationary point (a point with $0 \in \partial h(y)$). Such an function is given in Example 10.3.6.

Example 10.3.6 *The contours of the function $\phi(x) = h(c(x))$ where $h : \mathbb{R}^3 \to \mathbb{R}$ and $c : \mathbb{R}^2 \to \mathbb{R}^3$ are defined by*

$$c(x) = \begin{bmatrix} -5x_1 + x_2 \\ x_1^2 + x_2^2 + 4x_2 \\ 5x_1 + x_2 \end{bmatrix} \quad \text{and } h(c) = \max_{i=1,\ldots,3} c_i$$

are sketched in Figure 10.3.1. Starting from $\mathbf{x}^{(1)} = [\ 2 \quad \frac{-3+\sqrt{33}}{2}\ ]^T$ the steepest descent method converges to $\bar{\mathbf{x}} = 0$ which is not a stationary point. The minimizer is $\mathbf{x}^* = [\ 0 \quad -3\ ]^T$. The reason for this false convergence is that at every iterate $\mathbf{x}^{(k)}$ only the subdifferentials produced by c_1, c_2 or c_2, c_3 are used. At the limit point $\bar{\mathbf{x}}$ all 3 component functions are active and contribute to the subdifferential.

A general approach to nonsmooth optimization is *bundle methods* where a single subgradient $u^{(k)} \in \partial\phi^{(k)}$ at each iterate $\mathbf{x}^{(k)}$ is used to build an approximation to the generalized subdifferential. If it is only possible to calculate a single element of the generalized subdifferential at a point then bundle methods should be considered. Bundle methods are at best linearly convergent. However in convex composite optimization it is often possible to exploit the structure of the problem to produce more efficient and robust numerical methods.

Consider the composite function

$$\phi(\mathbf{x}) = f(\mathbf{x}) + h(c(\mathbf{x})).$$

At a point $\mathbf{x}^{(k)}$ linearize the functions f and c by

$$\begin{aligned}
c(\mathbf{x} + \mathbf{d}) &= c^{(k)} + A^{(k)^T}\mathbf{d} + o(||\mathbf{d}||) \\
f(\mathbf{x} + \mathbf{d}) &= f^{(k)} + \nabla f^{(k)^T}\mathbf{d} + o(||\mathbf{d}||).
\end{aligned}$$

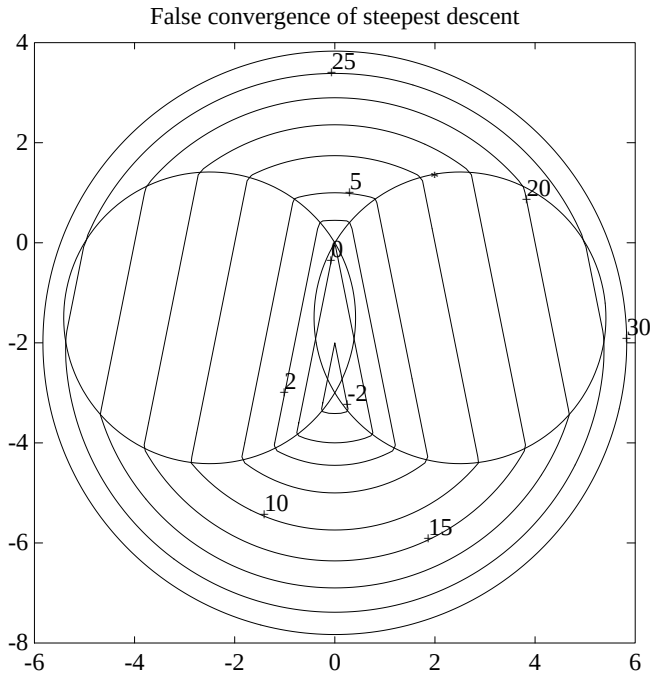


Figure 10.3.1: False convergence of steepest descent method

Using these linearizations, but keeping h exact, produces the subproblem

$$\begin{array}{ll} Minimize & f^{(k)} + \nabla f^{(k)^T}\mathbf{d} + h(c^{(k)} + A^{(k)^T}\mathbf{d}). \\ \mathbf{d} \in \mathbb{R}^n & \end{array} \tag{10.3.14}$$

Subproblem (10.3.14) requires the minimization of the function h of an affine function of the variables $\mathbf{d}$. This can be done efficiently when h is a polyhedral convex function (2.2.8). Then (10.3.14) is equivalent to

$$\begin{array}{ll} Minimize & \nabla f^{(k)^T}\mathbf{d} + \nu \\ \mathbf{d} \in \mathbb{R}^n,\ \nu \in \mathbb{R} & \\ Subject\ to & \nu - h_i^T A^{(k)^T}\mathbf{d} \geq h_i^T c^{(k)} + \beta_i \quad i = 1, \ldots, \ell. \end{array}$$

This is a linear programming subproblem in $n+1$ variables, but with ℓ general linear constraints. For some polyhedral function h, particularly when h is derived from the 1-norm, more efficient methods can be developed for minimizing the subproblem (10.3.14).

To produce methods with superlinear or better rates of convergence curvature information must be used. The second order optimality conditions indicate that the appropriate curvature term is the Hessian of the Lagrangian function. This produces the subproblem

$$\begin{array}{ll} Minimize & f^{(k)} + \nabla f^{(k)^T}\mathbf{d} + \mathbf{d}^T B^{(k)}\mathbf{d} + h(c^{(k)} + A^{(k)^T}\mathbf{d}) \\ \mathbf{d} \in \mathbb{R}^n & \end{array} \tag{10.3.15}$$

where $B^{(k)}$ is the Hessian $\mathcal{L}(\mathbf{x}^{(k)}; \lambda^{(k)})$ or some approximation to it. If h is a polyhedral convex function then (10.3.15) is equivalent to

$$\begin{array}{ll} Minimize & \nabla f^{(k)^T}\mathbf{d} + \mathbf{d}^T B^{(k)}\mathbf{d} + \nu \\ \mathbf{d} \in \mathbb{R}^n,\ \nu \in \mathbb{R} & \\ Subject\ to & \nu - h_i^T A^{(k)^T}\mathbf{d} \geq h_i^T c^{(k)} + \beta_i \quad i = 1, \ldots, \ell \end{array}$$

This is a quadratic programming subproblem. Again it is possible to develop numerical methods for solving (10.3.15) that exploit the structure of h rather than treating h as a general polyhedral convex function.

10.3.5 Linear ℓ_1

Consider the case where $c : \mathbb{R}^n \to \mathbb{R}^m$ is affine and $h = ||c||_1$, so $c(\mathbf{x}) = A^T\mathbf{x} + b$ where $A \in \mathbb{R}^{n\times m}$ has columns $\mathbf{a}_i$ for $i = 1, \ldots, m$, and

$$\phi(x) \equiv h(c(x)) = \sum_{i=1}^{m} |a_i^T x + b_i|.$$

As c is affine the composite function ϕ is convex. The subdifferential of $\phi(x)$ is

$$\partial\phi(x) = \left\{ u \in \mathbb{R}^m : u = \sum_{i:|c_i(x)|>0} sign(c_i(x))a_i + \sum_{i:c_i(x)=0} \lambda_i a_i \text{ where } |\lambda_i| \le 1 \right\}.$$

Define the set of active component functions

$$\mathcal{A}(\mathbf{x}) = \{\, i \in 1, \ldots, m : c_i(x) = 0 \,\}, \tag{10.3.16}$$

so $i \notin \mathcal{A}(\mathbf{x}) \iff c_i(x) \neq 0$. Let

$$w(x) = \sum_{i\notin\mathcal{A}(\mathbf{x})} sign(c_i(x))a_i. \tag{10.3.17}$$

Let $t(x) = |\mathcal{A}(\mathbf{x})|$ and let $V(x)$ be the n by $t(x)$ matrix whose columns are the vectors a_i for $i \in \mathcal{A}(x)$, so

$$V(x) = [\, a_i : i \in \mathcal{A}(\mathbf{x}) \,] \tag{10.3.18}$$

Then $w(x)$ is the gradient of the smooth part of $\phi(x)$ and

$$\partial\phi(x) = \{\, u \in \mathbb{R}^m : u = w(x) + V(x)v \text{ where } |v_i| \le 1 \; i = 1, \ldots, t(x) \,\} \tag{10.3.19}$$

The steepest descent direction at a point $\mathbf{x}^{(k)}$ comes from finding the point closest to the origin in $\partial\phi(\mathbf{x}^{(k)})$. Let $v^{(k)}$ solve

$$\begin{array}{ll} Minimize & ||w^{(k)} + V^{(k)}v||_2 \\ v \in \mathbb{R}^{t^{(k)}} & \\ Subject\ to & -1 \le v_i \le 1 \quad i = 1, \ldots, t(x). \end{array}$$

This is a least squares problem with simple lower and upper bounds on the variables. This least squares problem still involves a combinatorial aspect as it is not known which (if any) of the simple lower and upper bounds are active at a solution. Efficient numerical linear algebra techniques, such as the QR factorization and updating the QR factorization when bounds on the variables are added or dropped are required to solve this subproblem efficiently.

The steepest descent direction at x is then

$$\mathbf{d}^{(k)} = -(w^{(k)} + V^{(k)}v^{(k)})$$

If $\mathbf{d}^{(k)} = 0$ then $0 \in \partial\phi(\mathbf{x}^{(k)})$ and $\mathbf{x}^{(k)}$ is a minimizer of $\phi(\mathbf{x})$.

If $0 \notin \partial\phi(\mathbf{x}^{(k)})$ then $\mathbf{d}^{(k)}$ is a descent direction for ϕ at $\mathbf{x}^{(k)}$. A line search method would then find

$$\alpha^{(k)} = \arg\min_{\alpha\ge 0} \phi(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$$

and set

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}.$$

Consider the function

$$\ell_k(\alpha) \equiv \phi(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) = \sum_{i=1}^{m} |\alpha a_i^T \mathbf{d}^{(k)} + a_i^T \mathbf{x}^{(k)} + b_i| \tag{10.3.20}$$

Let $p_i = a_i^T \mathbf{d}^{(k)}$ and $q_i = a_i^T \mathbf{x}^{(k)} + b_i$, so

$$\ell_k(\alpha) = \sum_{i=1}^{m} |p_i + \alpha q_i|.$$

The univariate function ℓ_k is bounded below so it has a finite minimum. Moreover as ℓ_k is a piecewsie linear function the minimizer $\alpha^{(k)}$ must occur at one of the points

$$\alpha_i = \frac{-p_i}{q_i} \text{ where } q_i \neq 0.$$

Partial sorting algorithms can be used to find a minimizer by finding the first $\alpha_i > 0$ satisfying

$$\ell_k'(\alpha_{i-1}; 1) < 0 \text{ and } \ell_k'(\alpha_i; 1) \geq 0$$

efficiently (in particular more efficiently than calculating the value of $\ell_k(\alpha_i)$ for all i).

10.3.6 Exercises

1. Let $h : \mathbb{R}^m \to \mathbb{R}$ be a function for which the directional derivative $h'(y; d)$ exists and is finite for all y and d. Let
$$\Psi(y) = \{\, u \in \mathbb{R}^m : h'(y; d) \geq u^T d \quad \forall d \,\}.$$
   (a) Show that $\Psi(y)$ is a convex set for any y.
   (b) Let $m = 1, h(y) = -|y|$ and $\bar{y} = 0$.
      i. Calculate $h'(\bar{y}; d)$ for any $d \in \mathbb{R}$.
      ii. Calculate $\Psi(\bar{y})$.
      iii. Plot $h(y)$ and geometrically explain your value 0f $\Psi(\bar{y})$.

2. Let $n = 2$, $m = 2$, $\phi(\mathbf{x}) = h(c(\mathbf{x}))$
$$h(c) = \max\{\, c_1, c_2 \,\} \text{ and } c(\mathbf{x}) = \begin{bmatrix} 2x_1^2 + 2x_2^2 + 4x_2 \\ x_1^2 + x_2^2 - 2x_2 \end{bmatrix}.$$
   Let $\mathbf{x}^* = 0$. The contours of $\phi(\mathbf{x})$ are sketched in Figure 10.3.2.
   (a) Calculate $\partial h(c^*)$ and $\partial \phi(\mathbf{x}^*)$.
   (b) Verify that the first order necessary conditions are satisfied at $\mathbf{x}^*$.
   (c) Are the first order sufficeint conditions satisfied at $\mathbf{x}^*$?
   (d) Calculate the Lagrangian function $\mathcal{L}(\mathbf{x}; \lambda)$ and the Hessian $\nabla_{\mathbf{x}}^2 \mathcal{L}(\mathbf{x}; \lambda)$.
   (e) Calculate the set of directions $\mathcal{D}(\mathbf{x}^*)$ with zero directional derivative at $\mathbf{x}^*$.
   (f) Verify that the second order necessary conditions are satisfied at $\mathbf{x}^*$.
   (g) Verfiy that the second order sufficient conditions are satisfied at $\mathbf{x}^*$.
   (h) Is $\phi(\mathbf{x})$ convex?
   (i) Is it necessary to calulcate $\mathcal{D}(\mathbf{x}^*)$ to verfiy the second order conditions?

3. Consider the function
$$\phi(\mathbf{x}) = \max\{\, -5x_1 + x_2,\ x_1^2 + x_2^2 + 4x_2,\ 5x_1 + x_2 \,\}$$
   discussed in Example 10.3.6.

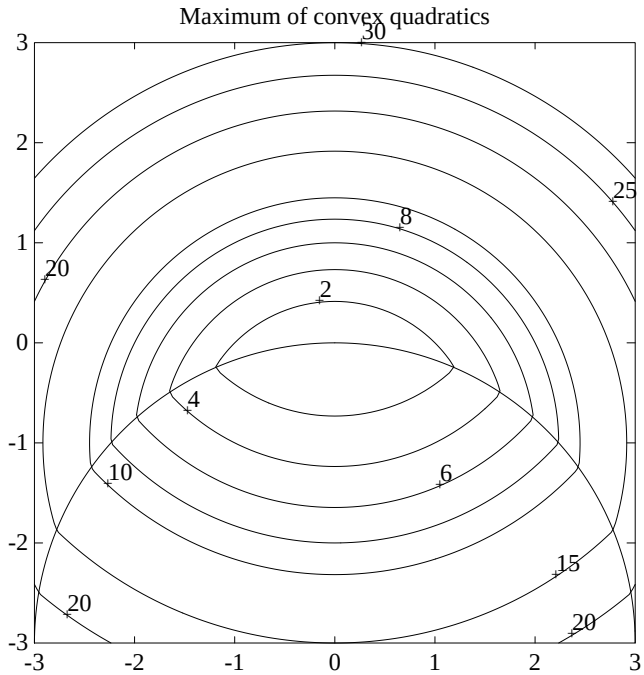


Figure 10.3.2: Maximum of convex quadratics

(a) Show that $\phi(\mathbf{x})$ is a convex function.

(b) At the point $\mathbf{x}^{(1)} = [\ 2 \quad \frac{-3+\sqrt{33}}{2}\ ]^T$.

  i. Calculate the subdifferential $\partial\phi(\mathbf{x}^{(1)})$.

  ii. Calculate the steepest descent direction $\mathbf{d}^{(1)}$.

  iii. Using an exact line search find $\alpha^{(1)}$ and hence $\mathbf{x}^{(2)}$.

(c) Calculate the subdifferential at $\bar{\mathbf{x}} = 0$ and show that $\bar{\mathbf{x}}$ is not a stationary point.

(d) Calculate the subdifferential at $\mathbf{x}^* = [0\ -3]^T$ and show that $\mathbf{x}^*$ is the strict minimizer of $\phi(\mathbf{x})$.

10.4 References

1. F. H. Clarke, *Optimization and Nonsmooth Analysis* (John Wiley, 1983).

2. R. Fletcher, *Practical Methods of Optimization*, (John Wiley, 2nd Edition, 1987).

3. H. D. Ioffe and V. M. Tihomirov, *Theory of Extremal Problems*, (North-Holland, Amsterdam, 1979).

4. K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, (Lecture Notes in Mathematics 1133, Springer, Berlin, 1985).

5. M. R. Osborne, *Finite Algorithms in Optimization and Data Analysis* (John Wiley, Chichester, 1985).

6. R. T. Rockafellar, *Convex Analysis*, (Princeton University Press, 1970).

Chapter 11

Probabilistic Methods and Global Optimization [H]

Problems in which there may be many local minima, but a global minimum is required (see [Gro96] for example), pose significant extra difficulties. This is especially true if the objective function has a low degree of continuity, for example it it may be discontinuous or have derivative discontinuities. In these situations classical methods which assume the objective is twice continuously differentiable, cannot be used to produce even a local minimizer.

This chapter will present an introduction to three approaches to finding a global optimization. These are

- The use of *quasi-Monte Carlo methods*, in which a fixed, well chosen, set of points is used to explore the feasible region. A good high-level reference is the book by Niederreiter [Nie92].
- The use of *simulated annealing*, in which the objective function is allowed to increase with a probability e^{-Kt} which decreases as temperature t increases. The aim is to allow a good chance that the method will escape from a basin of attraction on the early iterations. Simulated annealing was initially developed in the early 1980s [KGV83]. Practical difficulties are the rate at which the temperature should decrease for a particular problem. Some of these difficulties are overcome by *adaptive simulated annealing* (ASA) [CMMR87, Ing89, IR92].
- The use of *Genetic algorithms*, in which a population evolves by rules analogous to those governing natural selection. Preference is given to the fittest individuals (those with the lowest function value in a minimization problem) surviving and reproducing. New individuals (points) are created by *crossover* and *mutation* of individuals from the previous generation. Genetic algorithms developed from the work of Holland [Hol75] on natural selection. Although genetic algorithms are normally phrased in terms of a binary encoding of the variables [Gol89, Dav91], this chapter only considers floating point number representations [Mic94] and their use in function minimization.

These global optimization techniques are often applied to combinatorial problems, like the travelling salesman problem. However here the interest is in their use for the global minimization of continuous functions with many local minima.

There have also been many recent theoretical advances in the area of global optimization and many new algorithms have been proposed (see for example [HP95, HPT95, HT90, RT89]).

11.1 Quasi Monte-Carlo methods

11.2 Simulated Annealing

11.3 Genetic Algorithms

Chapter 12

Stochastic Optimization [H]

12.1 Introduction

Stochastic optimization deals with optimization problems in which some of the problem data is uncertain. A typical example is the constraints specifying the demand for a product, where the demand in not know exactly, but perhaps the distribution of the demand is known. This produces constraints in which the right-hand-side of the constraints is a random variable representing the demand. Another example is that of maximizing the return on a portfolio (as considered in Chapter 8) where the return on an asset class in not known with any certainty.

Good general references on stochastic programming are Kall and Wallace [KW94], Prékopa [Pre95] and Birge and Louveaux [BL97]. The most widely studied area of stochastic programming is stochastic linear programming, starting with the early work of Dantzig [Dan59] in 1954. The importance of stochastic programming has long been recognised, but it is only recently that the computing power has become available to tackle practical stochastic programming problem. Wets [Wet96] discusses the challenges facing stochastic pgramming, and the need for efficient robust methods that can be developed into reliable software. Stochastic programming problems are typically solved either as large sparse (linear) programming problems in which the structure of the problem must be exploited to decompose it, or by some form of stochastic decomposition [HS96].

The Chapter assumes a background in Statistics as covered in Appendix B and the optimization theory and methods covered in Chapters 1 to 7.

Appendix A

Matrix Algebra

This appendix summarizes some of the matrix algebra which is important in the development, analysis and implementation of numerical methods. They are based on the book *Matrix Computations* by G. H. Golub and C. F. Van Loan [GV96], which is an excellent reference book on numerical linear algebra. A slower-paced introduction to matrix algebra is *Linear Algebra and its Applications* by G. Strang [Str88]. The efficient implementation of Fortran programs for numerical linear algebra is discussed in the LAPACK User's Guide [ABB+95] (which supersedes LINPACK [DBMS79] and EISPACK [SBD+67, GBDM77] for eigenvalue calculations). Complementing these Fortran routines are the extremely useful and powerful interactive matrix calculation packages such as MATLAB [Mat01, HH00, Pra01, SD01]. Simple examples in MATLAB are given throughout this appendix. For more details see the MATLAB or one of the texts on MATLAB. In MATLAB typing `help` gives a list of topics on which help is available, while `help command` will give information about `command`. More detailed information can be obtained through the coomand (`helpdesk` in MATLAB which opens a web browser infterface.

Most matrix calculations are performed in floating point arithmetic. It is essential that the accuracy of the calculations and the magnitudes of the numbers that can be represented is born in mind. The accuracy is typically measured by the *floating point relative accuracy* which is the smallest positive number ϵ such that

$$1 + \epsilon > 1$$

in the floating point arithmetic being used. In MATLAB the command

```
eps
```

gives the value for ϵ on the current system. The IEEE standard for floating point arithmetic also uses `Inf` for a number which is too large to be represented, and `Nan` for a quantity that is *not a number*. In MATLAB try the commands

```
% Anything on a line after a % is a comment
1 + eps > 1     % 1 is used to represent a true answer
1 + eps/2 > 1   % 0 is used to represent a false answer
-1 / 0          % Divisions by 0 produces a warning and an Inf
0 / 0           % Nan is used when the result is not defined
Inf - Inf       % Nan is used when the result is not defined
```

Notation: '$\iff$' is used as an abbreviation for 'if and only if'.

A.1 Vectors and Matrices

The vector space of all n-dimensional *column* vectors with real elements is denoted by $\mathbb{R}^n$, so

$$\mathbf{a} \in \mathbb{R}^n \iff \mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

where $a_i \in \mathbb{R}$ for $i = 1, \ldots, n$. The vector space of all real m by n matrices is denoted by $\mathbb{R}^{m\times n}$, so

$$A \in \mathbb{R}^{m\times n} \iff A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix},$$

where $a_{ij} \in \mathbb{R}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Thus $\mathbb{R}^n$ can be identified with $\mathbb{R}^{n\times 1}$. Similar definitions apply for $\mathbb{C}^n$ the vector space of n-dimensional column vectors with complex entries, and $\mathbb{C}^{m\times n}$ the vector space of m by n matrices with complex entries. However these notes deal primarily with real vectors and matrices. In MATLAB the basic object is a matrix. The commands

```
a = [1:6]                  % Row vector with elements 1,2,3,4,5,6
b = [1; -2; 3; -4]         % Column vector
A = [1 2 3 4; 5 6 7 8];    % Semi-colon ; at the end of a line suppresses output
A                          % Display A
[m n] = size(A)
```

define a six element row vector (or 1 by 6 matrix) `a` with $a_i = i$, a four element column vector (or 4 by 1 matrix) `b` and a 2 by 4 matrix `A`. The square brackets `[]` define a matrix, with elements across a row separated by blanks or a comma `,` and rows separated by a new line or a semi-colon `;`. The command `size` gives the size of a matrix. Here the number of rows of `A` is stored in the variable `m` and the number of columns of `A` is stored in the variable `n`. MATLAB is case sensitive, so `a` and `A` are different. The semi-colon `;` at the end of a command suppresses output. Anything after a percentage sign `%` is treated as a comment, and is ignored by the MATLAB interpreter.

The basic operations with matrices are:

1. Addition ($\mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$)

$$C = A + B \iff c_{ij} = a_{ij} + b_{ij}, \quad \text{for } i = 1, \ldots, m \;\; j = 1, \ldots, n.$$

2. Multiplication by a scalar ($\mathbb{R} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$)

$$C = \alpha A \iff c_{ij} = \alpha a_{ij}, \quad \text{for } i = 1, \ldots, m \;\; j = 1, \ldots, n.$$

3. Matrix-matrix multiplication ($\mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{m\times p}$)

$$C = AB \iff c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad \text{for } i = 1, \ldots, m \;\; j = 1, \ldots, p.$$

4. Transposition ($\mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$)

$$C = A^T \iff c_{ij} = a_{ji}, \quad \text{for } i = 1, \ldots, n \;\; j = 1, \ldots, m.$$

An n-dimensional *row* vector is denoted by $\mathbf{a} = [\, a_1 \;\; \ldots \;\; a_n \,]^T$. Note that $(AB)^T = B^T A^T$. The MATLAB commands for these operations are

```
C = A + B        % Assumes A and B have the same size
C = alpha * A    % alpha is a scalar
C = A * B        % Assumes number of columns of A = number of rows of B
C = A'           % Transpose of A
```

Element by element multiplication of two matrices of the same size

$$c_{ij} = a_{ij} b_{ij} \qquad i = 1, \ldots, m \quad j = 1, \ldots, n$$

can be achieved by the MATLAB command

```
C = A .* B        % Element by element multiplication
```

The dimensions of matrices *must be* checked when doing matrix operations. Matrix multiplication is *not commutative*, that is if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ then $C = AB$ is defined but the product BA is not defined unless $m = p$. Even if $A, B \in \mathbb{R}^{n\times n}$ then, except for special cases, $AB \neq BA$. For example

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \Longrightarrow AB = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \neq BA = \begin{bmatrix} -2 & -2 \\ 2 & 2 \end{bmatrix}.$$

A typical matrix-vector product is $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ so

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \text{for } i = 1, \ldots, m,$$

which represents a system of m linear equations in n unknowns. The linear system can also be written as

$$\sum_{j=1}^{n} \mathbf{a}_j x_j = \mathbf{b}$$

where $\mathbf{a}_j \in \mathbb{R}^m$ is the jth column of A.

The *outer product* of $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$ is the m by n matrix

$$\mathbf{a}\mathbf{b}^T = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} b_1 & \ldots & b_n \end{bmatrix} = \begin{bmatrix} a_1b_1 & \cdots & a_1b_n \\ \vdots & \ddots & \vdots \\ a_mb_1 & \cdots & a_mb_n \end{bmatrix}.$$

In contrast if $n = m$ then the *inner product* (or dot product) of two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ is the *scalar* (or a 1 by 1 matrix)

$$\mathbf{a}^T\mathbf{b} = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \sum_{i=1}^{n} a_ib_i = \mathbf{b}^T\mathbf{a}.$$

As $\mathbf{a}^T\mathbf{b}$ is a scalar for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and transposition does not alter a scalar, it follows that $\mathbf{a}^T\mathbf{b} = \mathbf{b}^T\mathbf{a}$. For example if $\mathbf{a} = [\,1 \quad 2 \quad -2\,]^T$ and $\mathbf{b} = [\,-3 \quad 2 \quad -1\,]^T$ then

$$\mathbf{a}^T\mathbf{b} = \begin{bmatrix} 1 & 2 & -2 \end{bmatrix} \begin{bmatrix} -3 \\ 2 \\ -1 \end{bmatrix} = 3 = \mathbf{b}^T\mathbf{a},$$

while

$$\mathbf{a}\mathbf{b}^T = \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix} \begin{bmatrix} -3 & 2 & -1 \end{bmatrix} = \begin{bmatrix} -3 & 2 & -1 \\ -6 & 4 & -2 \\ 6 & 4 & 2 \end{bmatrix}.$$

In MATLAB this example is

```
a = [1; 2; -2]     % Define a column vector
b = [-3; 2; -1]
aTb = a' * b       % Inner product produces a scalar
bTa = b' * a       % Same as a'*b as result is a scalar
abT = a * b'       % Outer product produces a matrix
baT = b * a'       % Different matrix to a*b'
```

A *line* in $\mathbb{R}^n$ is conveniently parametrized by a non-zero direction vector $\mathbf{d} \in \mathbb{R}^n$ and a point $\mathbf{a} \in \mathbb{R}^n$ lying on the line. Thus

$$\{\, \mathbf{x}(\alpha) = \mathbf{a} + \alpha\mathbf{d} : \alpha \in \mathbb{R} \,\}$$

represents a line passing through the point $\mathbf{a} \in \mathbb{R}^n$ and parallel to the direction vector $\mathbf{d}$. The direction vector $\mathbf{d}$ can be multiplied by a nonzero constant without changing the line. A *ray* in $\mathbb{R}^n$ is parametrized as

$$\{\, \mathbf{x}(\alpha) = \mathbf{a} + \alpha \mathbf{d} : \alpha \geq 0, \alpha \in \mathbb{R} \,\}.$$

Another important geometrical object in $\mathbb{R}^n$ is a *hyperplane* defined by

$$H = \{\, \mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T \mathbf{x} = \beta \,\}.$$

Here $\mathbf{a} \in \mathbb{R}^n$ is the *normal* to the hyperplane and $\beta \in \mathbb{R}$ is determined by knowing a point through which the hyperplane passes. Associated with a hyperplane there are two *half-spaces*

$$H_+ = \{\, \mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T \mathbf{x} \geq \beta \,\}$$

and

$$H_- = \{\, \mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T \mathbf{x} \leq \beta \,\}.$$

The efficient implementation of basic matrix operations is discussed in Section A.9.

A.2 Special Matrices

Matrices with special patterns of zeros occur frequently. In particular A is

1. *Diagonal* $\iff a_{ij} = 0$ for all $i \neq j$.
2. *Upper triangular* $\iff a_{ij} = 0$ for all $i > j$.
3. *Unit upper triangular* $\iff a_{ij} = 0$ for all $i > j$ and $a_{ii} = 1$ for all i.
4. *Lower triangular* $\iff a_{ij} = 0$ for all $j > i$.
5. *Unit lower triangular* $\iff a_{ij} = 0$ for all $j > i$ and $a_{ii} = 1$ for all i.
6. *Banded* with lower bandwidth m_l and upper bandwidth m_u $\iff a_{ij} = 0$ for all $i - j > m_l$ and $j - i > m_u$. The total bandwidth is $m_l + m_u + 1$.
7. *Tridiagonal* $\iff a_{ij} = 0$ for all i, j with $|i - j| > 1$.
8. *Sparse* if it has relatively few nonzero elements.
9. *Toeplitz* $\iff a_{ij} = \alpha_{j-i}$.

A diagonal matrix is a banded matrix with $m_l = m_u = 0$. A tridiagonal matrix is a banded matrix with $m_l = m_u = 1$. A Toeplitz matrix is constant on bands about the main diagonal.

The sparsity of A is the number of non-zero elements in A divided by the total number of elements in A, usually expressed as a percentage.

Example A.2.1 (Discretization of Laplacian) *Large sparse matrices frequently arise from the discretization of partial differential equations. For example central difference approximations to the negative Laplacian*

$$-\frac{\partial^2 w(u,v)}{\partial u^2} - \frac{\partial^2 w(u,v)}{\partial v^2}$$

gives the matrix A whose non-zero elements are drawn in the first plot in Figure A.2.1. Figure A.2.1 also plots the the diagonal of A, the strict lower triangle of A, and the strict upper triangle of A. The matrix A has 364 non-zero elements out of a total of $80 \times 80 = 6,400$ elements. Thus A has sparsity of $364/6400 = 0.0569$ or 5.7%. The matrix A is banded with $m_l = m_u = 8$. The diagonal elements of A are all 4, and all non-zero off diagonal elements are -1. A is also symmetric, and positive definite (see page 228), with eigenvalues (see Section A.5) in the interval $[0.2016, 7.7984]$. MATLAB code for this example is in the file `laplace.m`.

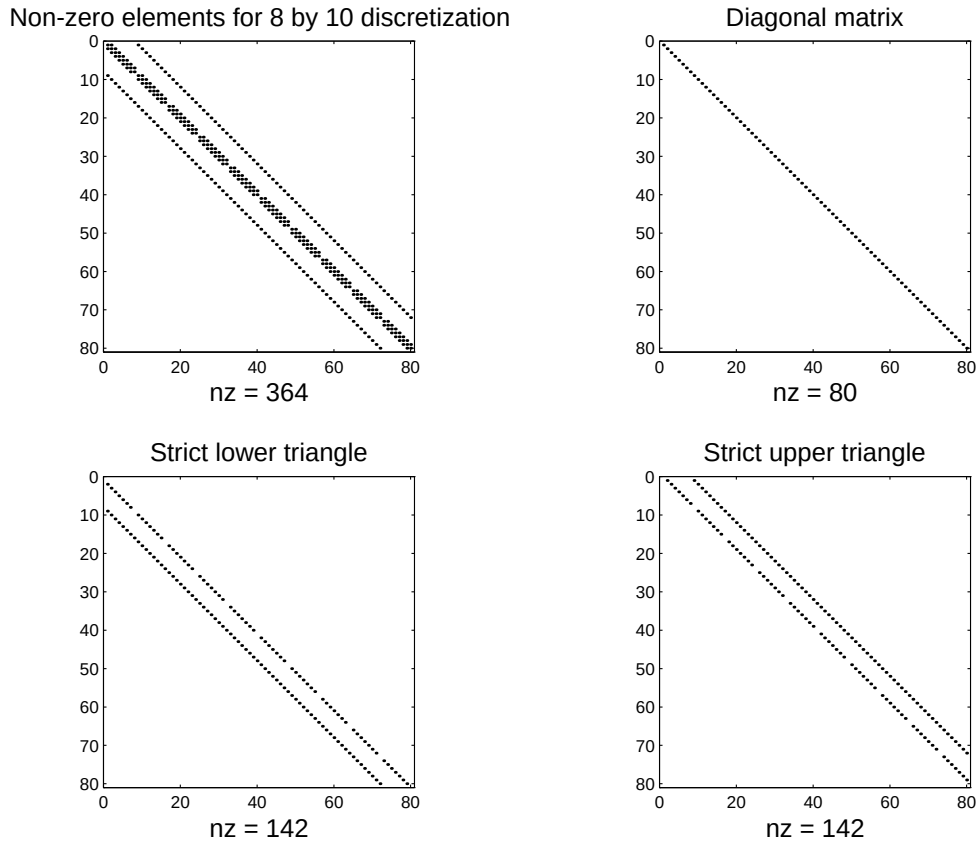


Figure A.2.1: Diagonal, lower triangle and upper triangle of sparse A

An n by n diagonal matrix A with $a_{ii} = \alpha_i$ for $i = 1, \ldots, n$ is denoted by

$$A = \mathrm{diag}(\alpha_1, \ldots, \alpha_n) = \begin{bmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{bmatrix}.$$

A diagonal matrix is an example of a sparse matrix, as at most n of its n^2 elements are nonzero. Note that pre-multiplication of a matrix $A \in \mathbb{R}^{m\times n}$ by a diagonal matrix $D \in \mathbb{R}^{m\times m}$ scales the ith row of A by d_{ii}. Alternatively post-multiplication of A by a diagonal matrix $D \in \mathbb{R}^{n\times n}$ scales the jth column of A by d_{jj}. Related MATLAB commands are

```
n = 10              % Define size
a = [1:n]           % a = 1 by n matrix with elements 1, 2, 3, ..., n
A = diag(a)         % A is n by n matrix with elements of a on its diagonal
B = rand(n ,n)      % n by n random matrix with elements in (0, 1)
b = diag(B)         % diag can also extract diagonal of a matrix, giving a vector
D = diag(diag(B))   % Diagonal matrix, containing diagonal of B
L = tril(B)         % Lower triangle of B
U = triu(B)         % Upper triangle of B
spy(U)              % Plot non-zero elements in U
nnz(U)              % Number of non-zero elements in U
```

An important diagonal matrix is the n by n *identity matrix* I_n which satisfies $I_n A = A$ for any $A \in \mathbb{R}^{n\times m}$ and $AI_n = A$ for any $A \in \mathbb{R}^{m\times n}$. Its ith column, the ith unit vector in $\mathbb{R}^n$, is denoted by $\mathbf{e}_i$, so

$$I_n = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$$

and

$$\mathbf{e}_i = [\, 0 \quad \cdots \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0 \,]^T$$

where the 1 is in the ith position. Usually the dimension of the identity matrix is determined by the context in which it is used. The vector $\mathbf{e} \in \mathbb{R}^n$ is used to denote the n-dimensional vector with each component equal to 1, so

$$\mathbf{e} = \sum_{i=1}^{n} \mathbf{e}_i = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Matlab commands to get these quantities are

```
% Assume m, n, i are appropriately defined integers
I = eye(n)          % n by n identity matrix
X = zeros(m, n)     % m by n zero matrix
Y = ones(m, n)      % m by n matrix of all ones
ei = I(:, i)        % i th column of matrix I
e = ones(n, 1)      % n by 1 matrix of all ones
```

If the columns (or rows) of the identity matrix are reordered you get a permutation matrix. Thus $A \in \mathbb{R}^{n\times n}$ is

1. *Permutation* $\iff A = [\, \mathbf{e}_{i_1} \quad \dots \quad \mathbf{e}_{i_n} \,]$ where $(i_1, \dots, i_n)$ is a permutation of $(1, \dots, n)$.

Orthogonal matrices, corresponding to rotations of the unit vectors, form a very useful and important class of matrices (see Section 4.2). The matrix $A \in \mathbb{R}^{m\times n}$ is

1. *Orthogonal* $\iff A^T A = I_n$.

The matrix A is said to be *square* if and only if $m = n$, so A has the same number of rows as columns. There are several important types of square matrices. $A \in \mathbb{R}^{n\times n}$ is

1. *Symmetric* $\iff A^T = A$.
2. *Skew-symmetric* $\iff A^T = -A$.
3. *Positive definite* $\iff \mathbf{x}^T A\mathbf{x} > 0$, for all $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$.
4. *Positive semi-definite* $\iff \mathbf{x}^T A\mathbf{x} \geq 0$, for all $\mathbf{x} \in \mathbb{R}^n$.
5. *Negative definite* $\iff \mathbf{x}^T A\mathbf{x} < 0$, for all $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$.
6. *Negative semi-definite* $\iff \mathbf{x}^T A\mathbf{x} \leq 0$, for all $\mathbf{x} \in \mathbb{R}^n$.
7. *Indefinite* $\iff$ there exist $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ such that $\mathbf{x}^T A\mathbf{x} < 0$ and $\mathbf{y}^T A\mathbf{y} > 0$.

A matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if and only if its symmetric part, $(A + A^T)/2$, is positive definite. Using the above definitions the set of positive semi-definite matrices includes all positive definite matrices, while the set of negative semi-definite matrices includes all negative definite matrices.

For a real symmetric matrix A the computationally most efficient way of determining if A is positive definite is to check if the Cholesky factorization $A = LL^T$ of A exists (see Section A.7.2).

If a symmetric matrix is *strictly diagonally dominant*, that is

$$a_{ii} > \sum_{j\neq i} |a_{ij}| \qquad \forall\ i = 1, \dots, n,$$

then A is positive definite. This provides a *sufficient* condition for positive definiteness that is easy to check. A matrix which is not strictly diagonally dominant may or may not be positive definite.

A.3 Inverses and Determinants

If $A, B \in \mathbb{R}^{n\times n}$ satisfy $AB = I = BA$, then B is the *inverse* of A and is denoted by A^{-1}. If A^{-1} exists then A is said to be *nonsingular*. If A^{-1} does not exist then A is *singular*. Note that A^{-T} denotes $(A^{-1})^T = (A^T)^{-1}$. Note also that $(AB)^T = B^T A^T$ and $(AB)^{-1} = B^{-1}A^{-1}$, so $(AB)^{-T} = A^{-T}B^{-T}$.

If $A \in \mathbb{R}^{1\times 1}$ then its *determinant* is defined by $det(A) = A$. For $A \in \mathbb{R}^{n\times n}$

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j}\, a_{ij} \det(M_{ij}) \quad \text{for any } i \in 1, \ldots, n$$

where M_{ij} is the $(n-1)$ by $(n-1)$ matrix obtained by deleting the ith row and the jth column of A. Useful properties of the determinant for $A, B \in \mathbb{R}^{n\times n}$ and $\alpha \in \mathbb{R}$ include

1. $\det(AB) = \det(A)\det(B)$.
2. $\det(A^T) = \det(A)$.
3. $\det(\alpha A) = \alpha^n \det(A)$.
4. $\det(A) \neq 0 \iff A$ is nonsingular.

If $A \in \mathbb{R}^{n\times n}$ is a diagonal, upper triangular or lower triangular matrix then

$$\det(A) = \prod_{i=1}^{n} a_{ii},$$

so a practical way of calculating the determinant is provided by the LU factorization (see Section 7.1). In MATLAB the determinant and inverse are calculated by

```
% Assume the square matrix A is defined
detA = det(A)
Ainv = inv(A)
```

For the 2 by 2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$det(A) = a_{11}a_{22} - a_{21}a_{12}$ and the inverse of A is given by

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}.$$

For matrices of size $n = 2$ or $n = 3$ it is reasonable to use the fact that

- A is positive definite $\iff$ all principal minors of A are positive

to determine if A is positive definite. The kth *principal minor* of A is the determinant of the leading k by k submatrix of A

$$M_k = det \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}. \tag{A.3.1}$$

Thus $M_1 = a_{11}$ and $M_n = \det A$. For even moderate values of n calculating all these determinants is computationally prohibitive. A is negative definite if the principal minors M_k have signs $(-1)^k$ for $k = 1, \ldots, n$. A matrix with $n \geq 2$ and all negative principal minors is indefinite.

A.4 Subspaces, Dimension and Rank

A set of vectors $\{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ in $\mathbb{R}^n$ is *linearly independent* if and only if

$$\sum_{i=1}^{m} \alpha_i \mathbf{a}_i = 0 \iff \alpha_1 = \ldots = \alpha_m = 0.$$

Otherwise a nontrivial linear combination of $\mathbf{a}_1, \ldots, \mathbf{a}_m$ is zero and $\{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ is said to be *linearly dependent*. A single vector $\mathbf{a} \in \mathbb{R}^n$ is linearly independent if and only if $\mathbf{a} \neq 0$. Any set of more that n vectors in $\mathbb{R}^n$ is linearly dependent.

A.4.1 Range and Null Spaces

A *subspace* of $\mathbb{R}^m$ is a nonempty subset $\mathbb{R}^m$ that is also a vector space. The set of all linear combinations of $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$ is a subspace referred to as the *span* of $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$:

$$\operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\} = \left\{ \mathbf{v} \in \mathbb{R}^n : \mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{a}_i \text{ where } \alpha_1, \ldots, \alpha_n \in \mathbb{R} \right\}.$$

If $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are linearly independent and $\mathbf{b} \in \operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, then $\mathbf{b}$ is a unique linear combination of $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

A *basis* for a subspace is a linearly independent set of vectors which spans the subspace. If $S \subset \mathbb{R}^m$ is a subspace then there exists a set of basis vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$, where $n \leq m$, such that $S = \operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$. All bases for a subspace S have the same number of elements. This number is the *dimension* of S and is denoted by $\dim(S)$.

There are two important subspaces associated with a matrix $A \in \mathbb{R}^{m \times n}$. The *range* or *image* of A is

$$\operatorname{im} A = R(A) = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = A\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^n\},$$

and the *null space* or *kernel* of A is

$$\ker A = N(A) = \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} = 0\}.$$

Note that

$$N(A^T) = \{\mathbf{y} \in \mathbb{R}^m : A^T\mathbf{y} = 0\} = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y}^T A = 0^T\}$$

and that $R(A) \cup N(A^T) = \mathbb{R}^m$.

If $A = [\mathbf{a}_1 \cdots \mathbf{a}_n]$ where $\mathbf{a}_i \in \mathbb{R}^m$ then $R(A) = \operatorname{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$. The *rank* of a matrix A is defined by

$$\operatorname{rank}(A) = \dim R(A).$$

Note that $\dim N(A)$ is often referred to as the *nullity* of A, and that $\dim N(A) + \operatorname{rank}(A) = n$ for any $A \in \mathbb{R}^{m \times n}$. As $\operatorname{rank}(A) = \operatorname{rank}(A^T)$, the rank of a matrix is the maximum number of linearly independent rows or columns of the matrix. If $A \in \mathbb{R}^{m \times n}$ where $m \leq n$ then A has *full rank* if the rows of A are linearly independent.

If $m = n$, so $A \in \mathbb{R}^{n \times n}$, the following statements are equivalent:

1. A is nonsingular.
2. $N(A) = \{\mathbf{0}\}$.
3. $\operatorname{rank}(A) = n$.

Numerically determining the rank of a matrix is difficult, as it involves determining when a floating point number is non-zero. The singular value decomposition (SVD) of Section A.7.4 provides a good way of numerically determining the rank of a matrix. The QR factorization (see Section A.7.3) provides a practical way of finding a basis for the null space of A. In MATLAB the related commands are

```
r = rank(A)     % number of numerically positive singular values
Q = null(A)     % columns of Q form orthonormal basis for null space of A
```

For example let

$$A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} 3 & 0 & -9 \\ 0 & 4 & 0 \\ -2 & 0 & 6 \end{bmatrix}.$$

As $a_3 = -3a_1$ the matrix A has only two linearly independent columns, so rank $(A) = 2$. Thus

$$R(A) = \text{span}\left\{ \begin{bmatrix} 3 \\ 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ 4 \\ 0 \end{bmatrix} \right\}$$

and

$$N(A) = \text{span}\left\{ \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} \right\},$$

so $\dim R(A) = 2$ and $\dim N(A) = 1$.

A.4.2 Orthogonality and Conjugacy

A set of vectors $\{ \mathbf{a}_1, \ldots, \mathbf{a}_n \}$ in $\mathbb{R}^m$ is

1. *Orthogonal* $\iff$ $\mathbf{a}_i^T \mathbf{a}_j = 0$ for all $i \neq j$.
2. *Orthonormal* $\iff$ $\mathbf{a}_i^T \mathbf{a}_j = 0$ for all $i \neq j$ and $\mathbf{a}_i^T \mathbf{a}_i = 1$ for all i.

If the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$ are nonzero and form an orthogonal set then that set is linearly independent.

The *orthogonal complement* S^o of a subspace $S \subset \mathbb{R}^m$ is

$$S^o = \{ \mathbf{x} \in \mathbb{R}^m : \mathbf{x}^T \mathbf{y} = 0 \text{ for all } \mathbf{y} \in S \}.$$

Note that $R(A)^o = N(A^T)$. The set of vectors $\{ \mathbf{v}_1, \ldots, \mathbf{v}_n \}$ forms an *orthonormal basis* for a subspace $S \subset \mathbb{R}^m$ if it is orthonormal and spans S. Thus the columns of an n by n orthogonal matrix form an orthonormal basis for $\mathbb{R}^n$.

A set of nonzero vectors $\{ \mathbf{v}_1, \ldots, \mathbf{v}_n \}$ in $\mathbb{R}^m$ is said to be *conjugate* with respect to a positive definite matrix $A \in \mathbb{R}^{m \times m}$ if and only if

$$\mathbf{v}_i^T A \mathbf{v}_j = 0 \text{ for all } i \neq j.$$

Thus orthogonality is simply the special case of conjugacy that occurs when $A = I_m$.

A.4.3 Rank-1 Matrices

Frequently matrices are modified by making a rank-1 change, which can be represented by

$$B = A + \mathbf{u}\mathbf{v}^T,$$

where $A, B \in \mathbb{R}^{m \times n}$, $\mathbf{u} \in \mathbb{R}^m$, and $\mathbf{v} \in \mathbb{R}^n$. Similarly if $A, B \in \mathbb{R}^{n \times n}$ are symmetric and $\mathbf{u} \in \mathbb{R}^n$ then

$$B = A + \mathbf{u}\mathbf{u}^T$$

represents a symmetric rank-1 change to A.

If $A \in \mathbb{R}^{n \times n}$ is nonsingular and $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ satisfy $\mathbf{v}^T A^{-1} \mathbf{u} \neq -1$ then the *Sherman-Morrison formula*

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1} \mathbf{u}},$$

gives a formula for the inverse of the modified matrix. A useful generalization of this is the *Sherman-Morrison-Woodbury formula*

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1}$$

where $A \in \mathbb{R}^{n \times n}$, $U, V \in \mathbb{R}^{n \times k}$, and both A and $(I + V^T A^{-1} U)$ are nonsingular.

A.5 Eigenvalues and Eigenvectors

The *eigenvalues* of a square matrix $A \in \mathbb{R}^{n\times n}$ are the n roots $\lambda_1, \ldots, \lambda_n$ of its *characteristic polynomial*

$$det(\lambda I - A) = 0.$$

For a real square matrix the eigenvalues are either real or occur in complex conjugate pairs. If A is symmetric ($A^T = A$) then all the eigenvalues are real. Some useful properties of eigenvalues are that

$$\begin{aligned} det(A) &= \prod_{i=1}^{n} \lambda_i, \\ trace(A) &\equiv \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i. \end{aligned}$$

Thus a square matrix is nonsingular if and only if *all* its eigenvalues are non-zero.

The *spectral radius* of the matrix A is

$$\rho(A) \equiv \max_{i=1,..,n} |\lambda_i|.$$

If $X \in \mathbb{R}^{n\times n}$ is nonsingular then A and $X^{-1}AX$ have the same eigenvalues. In particular if Q is a square orthogonal matrix ($Q^TQ = I$, so $Q^{-1} = Q^T$) then A and Q^TAQ have the same eigenvalues.

If λ is an eigenvalue of $A \in \mathbb{R}^{n\times n}$ then the nonzero vectors $\mathbf{v} \in \mathbb{C}^n$ (i.e. $\mathbf{v}$ is an n-dimensional vector with complex elements) satisfying

$$A\mathbf{v} = \lambda\mathbf{v}$$

are the *eigenvectors* of A corresponding to λ. Note that an eigenvector can be multiplied by an arbitrary nonzero scalar. Thus it is common to normalize eigenvectors so that $\mathbf{v}^T\mathbf{v} = 1$, i.e. they have length one. If $D = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and $V = [\ \mathbf{v}_1 \quad \cdots \quad \mathbf{v}_n\ ]^T$ then

$$AV = VD.$$

In MATLAB associated commands are

```
% Assume the square matrix A has been defined
ev = eig(A)        % eigenvalues of A
trace(A)            % trace of A = sum of diagonal elements of A
sum(diag(A))        % trace of A = sum of diagonal elements of A
sum(ev)             % trace of A = sum of eigenvalues of A
det(A)              % determinant of A
prod(ev)            % product of eigenvalues of A
rho = max(abs(ev)) % spectral radius of A
[V, D] = eig(A)     % matrix V of eigenvectors, diagonal matrix D of eigenvalues
```

A.5.1 Eigenvalues of Special Matrices

It is easy to find the eigenvalues for certain special classes of matrices. If $A \in \mathbb{R}^{n\times n}$ is a diagonal, upper triangular or lower triangular matrix then $\lambda_i = a_{ii}$ for $i = 1, \ldots, n$. If A is an orthogonal matrix then $|\lambda_i| = 1$ for $i = 1, \ldots, n$.

If $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix then all the eigenvalues are real, and there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ such that

$$Q^TAQ = diag(\lambda_1, \ldots, \lambda_n).$$

The ith column $\mathbf{q}_i$ of the matrix Q is the eigenvector corresponding to λ_i normalized so that $\mathbf{q}_i^T\mathbf{q}_i = 1$. Frequently the eigenvalues are ordered so that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$$

and Q can be chosen to achieve this ordering.

If A is a n by n nonsingular matrix, so $\lambda_i(A) \neq 0$ for $i = i, \ldots, n$, then

$$\lambda_i(A^{-1}) = \frac{1}{\lambda_i(A)} \quad i = 1, \ldots, n.$$

Let A and E be n by n symmetric matrices. The eigenvalues of $A + E$ cannot be obtained directly from those of A and E except in special cases. In general [GV96][Theorem 8.1.5]

$$\lambda_i(A) + \lambda_1(E) \leq \lambda_i(A + E) \leq \lambda_i(A) + \lambda_n(E) \quad i = 1; \ldots, n,$$

where the eigenvalues are ordered $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. If E is a scalar multiple of the idenitity, $E = \nu I$, $\nu \in \mathbb{R}$, then

$$\lambda_i(A + \nu I) = \lambda_i(A) + \nu \quad i = 1, \ldots, n.$$

For a symmetric matrix the eigenvalues provide a theoretically convenient way of characterizing the definiteness of the matrix:

- A is positive definite $\iff$ $\lambda_i > 0$ for all $i = 1, \ldots, n$.
- A is positive semi-definite $\iff$ $\lambda_i \geq 0$ for all $i = 1, \ldots, n$.
- A is negative definite $\iff$ $\lambda_i < 0$ for all $i = 1, \ldots, n$.
- A is negative semi-definite $\iff$ $\lambda_i \geq 0$ for all $i = 1, \ldots, n$.
- A is indefinite $\iff$ $\lambda_i > 0$ for some i and $\lambda_j < 0$ for some j.

A nonsingular positive semi-definite matrix is positive definite, while a nonsingular negative semi-definite matrix is negative definite.

A.6 Vector and Matrix Norms

If $\alpha \in \mathbb{R}$ then its magnitude (or absolute value) $|\alpha|$ is defined by

$$|\alpha| = \begin{cases} \alpha & \text{if } \alpha \geq 0; \\ -\alpha & \text{if } \alpha \leq 0. \end{cases}$$

However in $\mathbb{R}^n$ and $\mathbb{R}^{m \times n}$ there are several possible definitions of the magnitude of a quantity. These are referred to as vector norms and matrix norms and are denoted using $\|.\|$.

A.6.1 Vector Norms

A *vector norm* on $\mathbb{R}^n$ is a function $\|.\|$ from $\mathbb{R}^n$ to $\mathbb{R}$ satisfying

1. $\|\mathbf{x}\| \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$ and $\|\mathbf{x}\| = 0 \iff \mathbf{x} = 0$.
2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (Triangle inequality).
3. $\|\alpha \mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$ for all $\alpha \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$.

The most widely used vector norms are the p-norms defined by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

and in particular the

1. *1-norm*

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

2. *2-norm*

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} = \sqrt{\mathbf{x}^T\mathbf{x}}$$

3. ∞-*norm* or maximum norm

$$\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

Whenever there is no subscript on the $\|.\|$ the 2-norm is being used unless otherwise indicated. Note that $\|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x}$, and that the 2-norm satisfies the Cauchy-Schwarz inequality

$$|\mathbf{x}^T\mathbf{y}| \leq \|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2.$$

The 2-norm is invariant under orthogonal transformations, that is if $Q^TQ = I$ then $\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2$.

In MATLAB vector norms can be calculated by the commands

```
% Assume the vector x is defined
n2 = norm(x)                % default is the 2-norm, norm(x, 2)
n2 = sqrt(x'*x)             % Assumes x is a column vector
np = norm(x, p)             % p-norm for 1 <= p < Inf
np = sum(abs(x).^p)^(1/p)
ni = norm(x, 'inf')         % Infinity-norm
ni = max(abs(x))
```

Many MATLAB functions, like `abs`, take a vector or matrix as input, and produce the function applied to each element of the input. With a vector argument `sum` calculates the sum of all the elements in the vector, and `max` calculates the maximum elements in the vector. If the input is a matrix then `sum` and `max` produce row vectors, with the sum or maximum taken over each column of the matrix. The `.` operator means that the operation is performed on each element of the vector or matrix, so

```
z.^p
```

produces a vector or matrix of the same size as `z` with *each* elements raised to the power `p`.

A useful generalization of the 2-norm is the *A-norm* defined by

$$\|\mathbf{x}\|_A = (\mathbf{x}^T A\mathbf{x})^{\frac{1}{2}}$$

where A is a positive definite matrix. Some idea of the characteristics of these norms can be obtained from their unit balls

$$N = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq 1\}.$$

Example A.6.1 (Unit balls in $\mathbb{R}^2$) *The unit balls in R^2 for the 1-norm, 2-norm, ∞-norm, and A-norm for*

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

are sketched in Figure A.6.1, where

$$\begin{aligned}
N_1 &= \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_1 \leq 1\} = \{\mathbf{x} \in \mathbb{R}^2 : |x_1| + |x_2| \leq 1\} \\
N_2 &= \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_2 \leq 1\} = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\} \\
N_\infty &= \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_\infty \leq 1\} = \{\mathbf{x} \in \mathbb{R}^2 : |x_1| \leq 1, |x_2| \leq 1\} \\
N_A &= \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_A \leq 1\} = \{\mathbf{x} \in \mathbb{R}^2 : 2x_1^2 + 2x_1x_2 + 2x_2^2 \leq 1\}.
\end{aligned}$$

Note that although norms are continuous functions, the 1-norm and ∞-norms are not continuously differentiable. This is illustrated by the sharp corners on the unit balls for these norms.

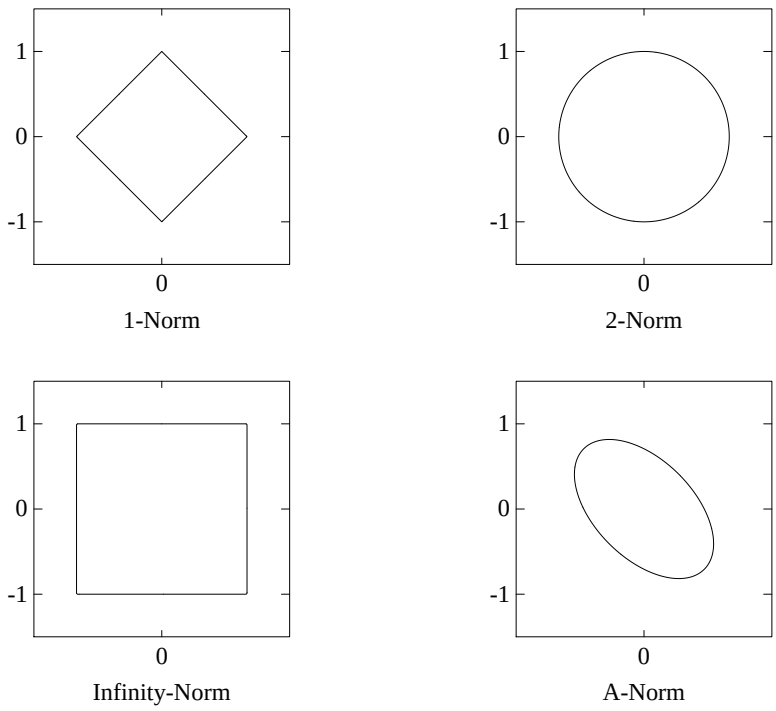


Figure A.6.1: Unit balls in $\mathbb{R}^2$

Norms provide the concept of distance between elements of $\mathbb{R}^n$ and are used in the definitions of continuity and convergence. For example a sequence $\{\mathbf{x}^{(k)}\}$ with $\mathbf{x}^{(k)} \in \mathbb{R}^n$ converges to $\mathbf{x}^* \in \mathbb{R}^n$ if and only if $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \to 0$ as $k \to \infty$.

Suppose $\hat{\mathbf{x}} \in \mathbb{R}^n$ is an approximation to $\mathbf{x} \in \mathbb{R}^n$. For a given norm the *absolute error* in $\hat{\mathbf{x}}$ is

$$\epsilon_a = \|\hat{\mathbf{x}} - \mathbf{x}\|$$

and if $\mathbf{x} \neq 0$ the *relative error* is

$$\epsilon_r = \frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|}.$$

The relative error using the ∞-norm gives some information about the number of correct significant digits in $\mathbf{x}$. In particular if

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \approx 10^{-p}$$

then the largest component of $\hat{\mathbf{x}}$ has approximately p correct significant digits. For example if $\mathbf{x} = [\,-3.641 \quad 0.7843\,]^T$ and $\hat{\mathbf{x}} = [\,-3.633 \quad 0.7915\,]^T$, then $\|\hat{\mathbf{x}} - \mathbf{x}\|/\|\mathbf{x}\|_\infty \approx 0.002 < 0.5 \times 10^{-2}$. From a direct comparison $\hat{x}_1$ has two correct significant digits while $\hat{x}_2$ has only one correct significant digit.

A.6.2 Matrix Norms

A *matrix norm* is a scalar function $\|.\|$ defined on $\mathbb{R}^{m\times n}$ satisfying

1. $\|A\| \geq 0$ for all $A \in \mathbb{R}^{m\times n}$ and $\|A\| = 0 \iff A = 0$.
2. $\|A + B\| \leq \|A\| + \|B\|$ for all $A, B \in \mathbb{R}^{m\times n}$ (Triangle inequality).
3. $\|\alpha\, A\| = |\alpha|\ \|A\|$ for all $\alpha \in \mathbb{R}$ and $A \in \mathbb{R}^{m\times n}$.

Matrix norms are *consistent* if and only if

$$\|AB\| \leq \|A\|\|B\| \text{ for all } A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times q}.$$

The most frequently used matrix norms are the *subordinate* matrix norms, which are defined in terms of vector norms by

$$\|A\|_p = \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}.$$

These include

$$\begin{aligned}
\|A\|_1 &\equiv \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_1}{\|\mathbf{x}\|_1} = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| \\
\|A\|_2 &\equiv \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\
\|A\|_\infty &\equiv \max_{x\neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}|.
\end{aligned}$$

If $A \in \mathbb{R}^{m\times n}$ where $m \neq n$ then the 2-norm is the largest singular value of A (see Section A.7.4. If $A \in \mathbb{R}^{n\times n}$ then

$$\|A\|_2 = \rho(A) \equiv \max_{i=1,\ldots,n} |\lambda_i|.$$

The subordinate matrix norms are consistent and have the important property that for every $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$

$$\|Ax\|_p \leq \|A\|_p \, \|\mathbf{x}\|_p.$$

Another useful matrix norm is the *Frobenius* norm

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}.$$

Note that the Frobenius norm is different from the matrix 2-norm even though its definition looks similar to the vector 2-norm. In MATLAB matrix norms can be calculated by the commands

```
% Assume the matrix A is defined
norm(A)          % default is the 2-norm, norm(A, 2)
norm(A, 1)       % 1-norm, maximum column sum, max(sum(abs(A)))
norm(A, 'inf')   % Infinity-norm, maximum row sum, max(sum(abs(A')))
norm(A, 'fro')   % Frobenius norm, sqrt(sum(sum(A.*A)))
```

For $A \in \mathbb{R}^{n\times n}$ these norms are related by

$$\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty} \leq \sqrt{n}\|A\|_2$$

and

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2.$$

Both the Frobenius norm and the 2-norm are invariant with respect to orthogonal transformations. That is if $Y \in \mathbb{R}^{p\times m}$ and $Z \in \mathbb{R}^{n\times q}$ satisfy $Y^TY = I_m$ and $Z^TZ = I_q$ then $\|YAZ\|_F = \|A\|_F$ and $\|YAZ\|_2 = \|A\|_2$.

A.7 Matrix Factorizations

Matrix factorizations are fundamental for the practical solution of systems of linear equations. Two important matrix factorizations are the LU factorization and the QR factorization. Here attention is focussed on the LU factorization of a square matrix and the QR factorization of a rectangular matrix. Fortran routines for these factorizations are available in the LAPACK library[ABB+95]. Alternatively the factorizations can easily be calculated using matrix calculators like MATLAB.

A.7.1 LU Factorization

Let A_k denote the kth leading principal submatrix of $A \in \mathbb{R}^{n\times n}$, that is the k by k matrix formed by the elements a_{ij} for $i, j = 1, \ldots, k$. If A_k is nonsingular for $k = 1, \ldots, n$ then there exists a unit lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ such that

$$A = LU.$$

As L is unit lower triangular $det(L) = 1$, so

$$det(A) = det(U) = \prod_{i=1}^{n} U_{ii}.$$

Unfortunately there are some simple matrices that do not have an LU factorization. For example

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

does not have an LU factorization because its leading principal submatrix is singular. Even if a matrix has an LU factorization the process of calculating the matrices L and U may not be numerically stable. These difficulties are overcome by *pivoting* (reordering the rows and/or columns of A). The most widely used pivoting strategy, called *partial pivoting*, uses just a reordering of the rows of A. In matrix notation this is represented using a permutation matrix P, which is just the identity matrix with its columns reordered. Note that in a practical implementation of pivoting only a vector is used to store the reordering represented by P.

If $A \in \mathbb{R}^{n\times n}$ is nonsingular then there exists a permutation matrix $P \in \mathbb{R}^{n\times n}$, a unit lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ such that

$$PA = LU.$$

In the last example A is in fact the 2 by 2 permutation matrix obtained by swapping the first and second columns of the identity matrix. Note that pre-multiplication of A by a permutation matrix P, to produce PA, reorders the rows of A. Alternatively post-multiplication of A by P, to produce AP, reorders the columns of A.

An equivalent form of the LU factorization is $PA = LD\hat{U}$ where $L \in \mathbb{R}^{n\times n}$ is a unit lower triangular matrix, $\hat{U} \in \mathbb{R}^{n\times n}$ is a unit upper triangular matrix, and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix. Here D contains the diagonal elements of the matrix U, $\hat{U}$ has 1s on its diagonal and $U = D\hat{U}$, so the i th row of $\hat{U}$ is obtained by dividing the i th row of U by U_{ii} when $U_{ii} \neq 0$.

The process used to calculate the LU decomposition is called *Gaussian elimination* and is discussed in detail in any of the references. In MATLAB the LU factorization is obtained by

```
% L product of lower triangular and permutation matrices, U upper triangular
[L, U] = lu(A)      % A = L * U
% L Lower triangular, U upper triangular, P permutation matrix
[L, U, P] = lu(A)   % P * A = L * U
```

A.7.2 Cholesky Factorization

It is very common in optimization methods to have to solve a linear system $Ax = b$ where the matrix A is known to be symmetric. Symmetry can be used to reduce the storage requirements for A from n^2 elements to $n(n+1)/2$. (See the LAPACK User's Guide [ABB+95] for a detailed discussion of symmetric storage schemes.)

If $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix which has an LU factorization there exists a unit lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ such that

$$A = LDL^T.$$

Unfortunately this is not possible for all symmetric matrices, as the previous example of a matrix which does not have an LU factorization illustrates. If pivoting is introduced then both row and column reordering must be used to preserve symmetry. This produces the factorization

$$PAP^T = LDL^T$$

where $P \in \mathbb{R}^{n\times n}$ is a permutation matrix. However if A is a positive definite or negative definite symmetric matrix then pivoting is unnecessary.

A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if and only if there exists a unit lower triangular matrix $L \in \mathbb{R}^{n\times n}$ and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ with $d_{ii} > 0$ for $i = 1, \ldots, n$ such that

$$A = LDL^T.$$

As $d_{ii} > 0$ let

$$D^{\frac{1}{2}} = diag(\sqrt{d_{11}}, \ldots, \sqrt{d_{nn}})$$

and $\hat{L} = LD^{\frac{1}{2}}$. Then

$$A = \hat{L}\hat{L}^T,$$

which is known as the *Cholesky factorization* of the matrix A. The Cholesky factorization provides both a practical way of checking whether a symmetric matrix is positive definite and an efficient stable way of solving positive definite symmetric linear systems. In MATLAB the Cholesky factorization is obtained by

```
% If A is symmetric positive definite no pivoting is required
R = chol(A)       % A = R' * R, R upper triangular
% If A is not positive definite
[R, p] = chol(A)  % R'*R = A(1:q,1:q), q = p - 1
```

A.7.3 QR Factorization

The QR factorization provides a numerically stable way of solving linear equations and linear least squares problems (see Section 8.2). It also can be used to provide orthonormal bases for both the range and the null space of a matrix. Details of how the QR factorization can be calculated are available in any of the references.

If $A \in \mathbb{R}^{n\times n}$ is nonsingular then there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ and an nonsingular upper-triangular matrix $R \in \mathbb{R}^{n\times n}$ such that

$$A = QR.$$

As $Q^TQ = I$ it follows that $Q^TA = R$ so the linear system $Ax = b$ is equivalent to $Rx = Q^Tb$, which can be efficiently solved by back-substitution (see Section A.8.1).

If $A \in \mathbb{R}^{m\times n}$ where $m > n$ and $rank(A) = n$, then there exists an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and a nonsingular upper-triangular matrix $R \in \mathbb{R}^{n\times n}$ such that

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Numerical it is desirable (essential if A does not have full rank) to introduce *column pivoting*, which produces

$$Q^TAP = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where P is a permutation matrix reordering of the columns of A.

Let $Q = \begin{bmatrix} Y & Z \end{bmatrix}$ where $Y \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{\times m}(m-n)$, so Y is formed by the first n columns of Q and Z is formed by the remaining $m-n$ columns of Q. As Q is a square orthogonal matrix

$$Q^TQ = I_m = QQ^T.$$

Hence

$$Q^TQ = \begin{bmatrix} Y^T \\ Z^T \end{bmatrix} \begin{bmatrix} Y & Z \end{bmatrix} = \begin{bmatrix} Y^TY & Y^TZ \\ Z^TY & Z^TZ \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ 0 & I_{m-n} \end{bmatrix}.$$

Thus $Y^TY = I_n$, $Z^TZ = I_{m-n}$, so Y and Z are orthogonal matrices, and $Y^TZ = 0$ and $Z^TY = 0$. Note that as Y, Z are not square matrices $YY^T \neq I_m$ and $ZZ^T \neq I_m$. In fact

$$I_m = QQ^T = YY^T + ZZ^T.$$

The columns of Y provide an orthonormal basis for the range of A as

$$A = YR,$$

and YY^T provides an orthogonal projection onto the range of A. The columns of Z provide an orthonormal basis for the null space of A^T as

$$Z^TA = 0,$$

and ZZ^T provides an orthogonal projection onto the null space of A^T.

In MATLAB these quantities can be calculated by

```
A = [1; 1; 2]        % Define 3 by 1 matrix A
[m, n] = size(A)
r = rank(A)
[Q, R] = qr(A)       % Calculate QR factorization
Q' * Q               % Check Q'*Q = I, so Q is orthogonal
Y = Q(:,1:r)         % Basis for range space of A
Y' * Y               % r by r identity matrix
Y * Y'               % Not the identity as Y is not square
Z = Q(:,r+1:m)       % Basis for null space of A'
Z' * Z               % m-r by m-r identity matrix
Z * Z'               % Not the identity as Z is not square
```

A.7.4 Singular Value Decomposition (SVD)

For any $A \in \mathbb{R}^{m\times n}$ the Singular Value Decomposition (SVD) exits. Assuming that $m \geq n$ the the SVD is

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal ($U^TU = I_m, V^TV = I_n$) and $\Sigma \in \mathbb{R}^{n\times n}$ is the diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$ and the singular values σ_i are ordered

$$\sigma_1 \geq \cdots \geq \sigma_n \geq 0.$$

If $n > m$ then

$$A = U \begin{bmatrix} \Sigma & 0 \end{bmatrix} V^T$$

and $\Sigma \in \mathbb{R}^m$, $\Sigma = diag(\sigma_1, \ldots, \sigma_m)$. This section deals with the case $m \geq n$.

The singular values of A are the ordered square roots of the eigenvalues of A^TA, i.e.

$$\sigma_i(A) = \sqrt{\lambda_i(A^TA)}.$$

For an n by n *symmetric* matrix A the singular values are the ordered magnitudes of the eigenvalues of A.

The smallest non-zero singular value is one of the numerically best ways of determining the rank r of a matrix, so

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

The presence of rounding error still makes the process of determining the rank difficult (is $\sigma_r = 4 \times 10^{-14}$ zero or not?).

If A is full rank, so $r = \min(m, n)$, then the condition umber of A is

$$\kappa(A) = \frac{\sigma_1}{\sigma_r}.$$

An example in MATLAB is

```
format short e              % Display in scientific notation
A = [1 1; 0 10*eps; 0 0]    % Define 3 by 1 matrix A
[m, n] = size(A)
svd(A)                      % Singular values of A
[U, S, V] = svd(A)          % Orthogonal U, V, diagonal S: A = U*S*V'
```

A.7.5 Sparse factorizations

A problem with calculating a factorization of a sparse matrix is that non-zero elements are often created in positions which originally contained zeros. This is known as *fill-in*, and increases both the storage and computation required.

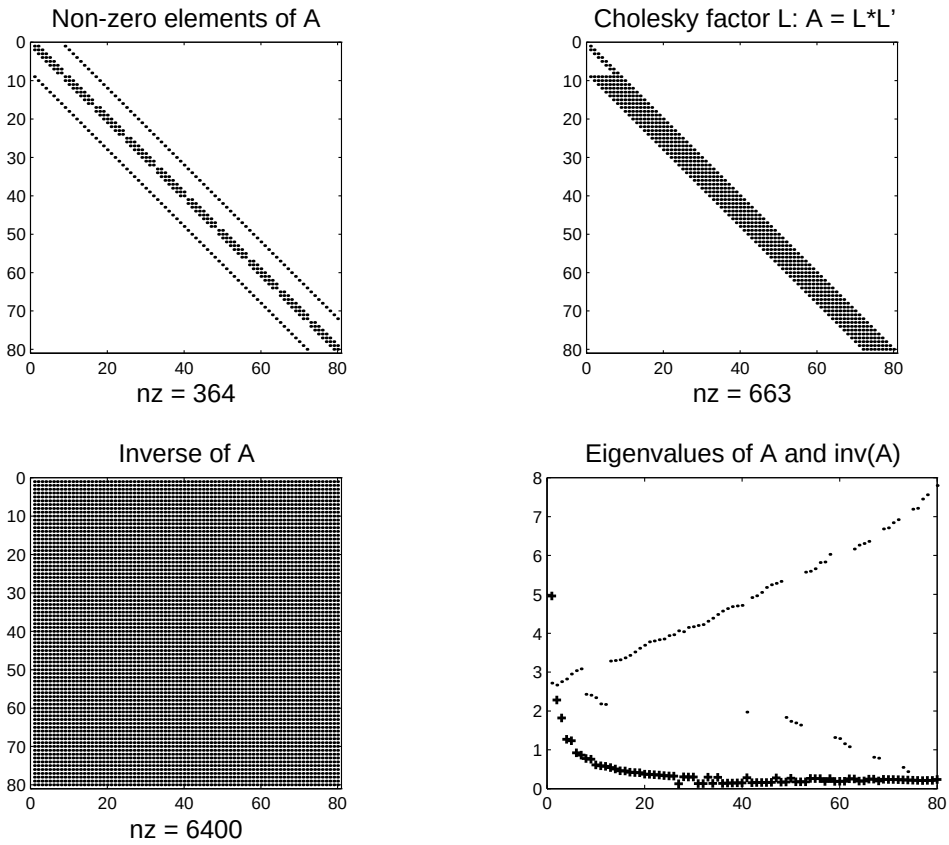


Figure A.7.1: Sparse A, Cholesky factor L, A^{-1}, and eigenvalues

Example A.7.1 (Discretization of Laplacian continued) *Consider the sparse matrix A in Example A.2.1 arising from the discretization of the negative Laplacian. As the matrix is symmetric and*

positive definite the Cholesky factorization can be used to solve the linear system $A\mathbf{x} = \mathbf{b}$. In A all the non-zeros occur on the diagonal and 4 bands. Figure A.7.1 shows the non-zero elements of the Cholesky factor L, in which fill-in has occurred, but only within the outermost bands of A. The fill-in is much worse in forming A^{-1}, which is a full matrix. The last plot in Figure A.7.1 gives the eigenvalues of A and A^{-1}.

A.8 Systems of Linear Equations

A frequently arising problem is to find an $\mathbf{x} \in \mathbb{R}^n$ satisfying the system of linear equations $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are known. Theoretically this problem has a solution if and only if $\mathbf{b} \in R(A)$, and it is unique if and only if $N(A) = \{0\}$. However we need practical methods that can efficiently be implemented on a computer. Moreover they must be numerically stable in the sense that the errors introduced by representing numbers in the finite precision of a computer must not be unduly amplified by the process of solving the linear system.

A.8.1 Triangular systems

Systems of linear equations in which the coefficient matrix is either lower or upper triangular are efficiently solved by by forward or back-substitution. If $L \in \mathbb{R}^{n\times n}$ is a nonsingular lower triangular matrix then the solution y to the linear system $L\mathbf{y} = \mathbf{b}$ is given by an algorithm known as *forward substitution*

$$\begin{aligned} y_1 &= b_1/L_{11} \\ y_i &= \left(b_i - \sum_{j=1}^{i-1} L_{ij}y_j\right)/L_{ii} \quad \text{for } i = 2,\ldots,n. \end{aligned}$$

Note that if L is a unit lower triangular matrix, so $L_{ii} = 1$ for all i, the forward substitution algorithm simplifies.

Similarly if U is a nonsingular upper triangular matrix the solution $\mathbf{x}$ to $U\mathbf{x} = \mathbf{y}$ is provided by an algorithm known as *back-substitution*

$$\begin{aligned} x_n &= y_n/U_{nn} \\ x_i &= \left(y_i - \sum_{j=i+1}^{n} U_{ij}y_j\right)/U_{ii} \quad \text{for } i = n-1,\ldots,1. \end{aligned}$$

A.8.2 Well-determined Systems

If $A \in \mathbb{R}^{n\times n}$ then $A\mathbf{x} = \mathbf{b}$ represents a system of n linear equations in n variables x. If the matrix A is nonsingular then $\mathbf{x} = A^{-1}b$. However instead of calculating the inverse A^{-1} it is much more efficient to calculate the LU factorization of A. Given a unit lower triangular matrix L, an upper triangular matrix U and a permutation matrix P such the $PA = LU$ the system

$$PA\mathbf{x} = LU\mathbf{x} = Pb$$

is solved by using forward substitution to solve $L\mathbf{y} = P\mathbf{b}$ for $\mathbf{y}$ and then using back-substitution to solve $U\mathbf{x} = \mathbf{y}$. Remember that $P\mathbf{b}$ is just a reordering of the elements of the vector $\mathbf{b}$.

If the matrix A is symmetric then the LDL^T factorization can be used to solve $A\mathbf{x} = \mathbf{b}$ by forward and back-substitution. One advantage of this is a saving in storage as the matrix U does not have to be used and an increase in speed if full use is made of the symmetric structure of A. Similarly if A is a symmetric positive definite matrix then the Cholesky factorization $A = \hat{L}\hat{L}^T$ provides an efficient stable way of solving the linear system $A\mathbf{x} = \mathbf{b}$.

In MATLAB the backslash \ is used to solve linear systems, for example

```
% Assume A is an n by n matrix and b is a column vector with n elements
x = A \ b     % solution of A*x = b using Gaussian elimination
```

See `help slash` for more details.

A.8.3 Numerical Sensitivity

It is important to know how the process of solving a system of linear equations affects the roundoff errors introduced by storing numbers in a computer. Some information can be obtained from the *condition number* $\kappa(A)$ of a square nonsingular matrix A, which is defined by

$$\kappa(A) = \|A\| \, \|A^{-1}\|,$$

where any matrix norm may be used. Note that for the 1-norm, 2-norm and ∞-norm (but not the Frobenius norm) $\|I\| = 1$, so $\kappa(A) \geq 1$. The condition number depends on the particular matrix norm used. For example when using the 2-norm

$$\kappa_2(A) = \|A\|_2 \, \|A^{-1}\|_2 = \frac{|\lambda_1|}{|\lambda_n|},$$

where $|\lambda_1| \geq, \ldots, \geq |\lambda_n|$ are the magnitudes of the eigenvalues of A. Hence a square orthogonal matrix Q has $\kappa_2(Q) = 1$, showing that orthogonal matrices are ideally conditioned. Even if the matrix is not square the 2-norm condition number is

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_r}$$

The invariance of the 2-norm and Frobenius norm to orthogonal transformations means that if R is obtained from a QR factorization of A (see section A.7.3) then

$$\kappa_2(A) = \kappa_2(R) \qquad \kappa_F(A) = \kappa_F(R).$$

Also if $\hat{L}$ comes from a Cholesky factorization of A (see section A.7.2 then

$$\kappa_2(A) = \kappa_2(L)^2.$$

Thus in these decompositions the condition number of A is obtainable from the triangular factor.

The condition number is a measure of the nonsingularity of a matrix. For a nearly singular matrix, with nearly linearly dependent columns, the condition number is correspondingly large. Badly conditioned matrices are those with large condition numbers. A precise measure of the sensitivity of a linear system $A\mathbf{x} = \mathbf{b}$ can be obtained by considering the parametrized system

$$(A + \epsilon\, C)\mathbf{x}(\epsilon) = \mathbf{b} + \epsilon\, \mathbf{c},$$

where $C \in \mathbb{R}^{n\times n}$, $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{x}(0) = \mathbf{x}$. If A is nonsingular then, using any vector norm and subordinate matrix norm,

$$\frac{\|\mathbf{x}(\epsilon) - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A)[\,\rho_A + \rho_b] + O(\epsilon^2),$$

where

$$\rho_A = \epsilon\,\frac{\|C\|}{\|A\|} \quad \text{and} \quad \rho_b = \epsilon\,\frac{\|\mathbf{c}\|}{\|\mathbf{b}\|}$$

represent the relative errors in A and b respectively. Thus the condition number gives an idea how the relative errors in A and $\mathbf{b}$ will be amplified when the linear system is solved for $\mathbf{x}$.

In practice the condition number is usually estimated rather than calculated exactly because of the computational cost of evaluating A^{-1}. The factorization routines in the LAPACK library [ABB+95] use an estimate (in fact a lower bound) on $1/\kappa_1(A)$. Other estimates and bounds are discussed in Golub and van Loan [GV96]. In MATLAB the condition number is obtained by

```
kappa = cond(A)    % 2-norm condition number
rc = rcond(A)      % Estimate of reciprocal of condition number using 1-norm
```

A classical example of an ill-conditioned matrix is the Hilbert matrix $H \in \mathbb{R}^{n\times n}$ defined by $H_{ij} = 1/(i+j+1)$ for $i, j = 1, \ldots, n$. The 10 by 10 Hilbert matrix has a condition number of 10^{13}, so if single precision arithmetic on a computer with a machine precision (the smallest positive number ϵ such that $1 + \epsilon > 1$) of around 10^{-7} is used to solve a linear system $H\mathbf{x} = \mathbf{b}$ then there will be NO correct digits in the answer. If double precision arithmetic with a precision of around 10^{-16} is used then there about 3 correct significant digits in the calculated value of $\mathbf{x}$. In MATLAB

```
H = hilb(10);       % 10 by 10 Hilbert matrix
cond(H)
condest(H)
```

A.8.4 Over-determined Linear Systems

If $A \in \mathbb{R}^{m\times n}$ where $m > n$ then $A\mathbf{x} = \mathbf{b}$ represents an over-determined system of m linear equations in n variables $\mathbf{x}$. If $\mathbf{b} \in R(A)$ then this system has a solution $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$, hence

$$A^T A\mathbf{x} = A^T\mathbf{b}.$$

If $rank(A) = n$ then $A^T A \in \mathbb{R}^{n\times n}$ is nonsingular, so

$$\mathbf{x} = (A^T A)^{-1} A^T b.$$

The matrix

$$A^+ = (A^T A)^{-1} A^T$$

is called the *pseudo inverse* or *Moore-Penrose generalized inverse* of A. If $m = n$ then $A^+ = A^{-1}$, however if $m > n$ then $A^+ A = I_n$ but $AA^+ \neq I_m$.

When $\mathbf{b} \notin R(A)$, so $A\mathbf{x} = \mathbf{b}$ has no solution, $\mathbf{x}^* = A^+\mathbf{b}$ is the solution to the linear *least squares problem*

$$\underset{\mathbf{x}\in\mathbb{R}^n}{Minimize} \quad \|A\mathbf{x} - \mathbf{b}\|_2^2 = \sum_{i=1}^m r_i^2 = \mathbf{r}^T\mathbf{r}$$

where $\mathbf{r} \in \mathbb{R}^m$ is the *residual*

$$\mathbf{r} = \mathbf{b} - A\mathbf{x}.$$

Then $\mathbf{b} \in R(A)$ if and only if $\mathbf{r}^* = \mathbf{b} - A\mathbf{x}^* = 0$, where $\mathbf{x}^*$ satisfies the *normal equations*

$$A^T A\mathbf{x}^* = A^T\mathbf{b}.$$

Example A.8.1 (Over-determined linear system) *Figure A.8.1 illustrates the over-determined lin-*

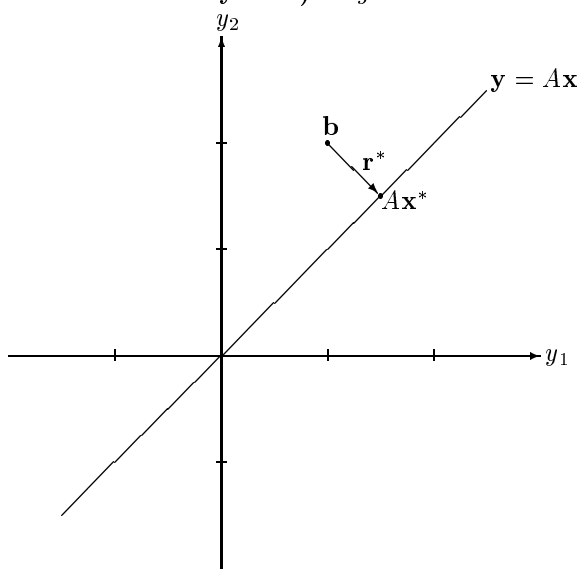


Figure A.8.1: Over-determined linear system, $\mathbf{x}^* = argmin\ \|A\mathbf{x} - \mathbf{b}\|_2$

ear system where $m = 2, n = 1$,

$$A = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad and \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

so

$$\mathbf{x}^* = 3/2 \quad \textit{and} \quad \mathbf{r}^* = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

In MATLAB, when $m > n$ the \ gives the least squares solution of the over-determined linear system. For example

```
A = [1; 1]
[m, n] = size(A)
b = [1; 2]
x = A \ b
r = b - A*x
```

Unfortunately forming $A^T A$ as in the normal equations is not a numerically stable process as it squares the conditioning of the problem. For example if

$$A = \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix} \quad \text{then} \quad A^T A = \begin{bmatrix} 1 & 1 \\ 1 & 1+\epsilon^2 \end{bmatrix}.$$

If ϵ is small enough so that $1 + \epsilon^2 = 1$ on a computer then $A^T A$ is numerically singular. However if $\epsilon \neq 0$ on the computer then A numerically has full rank. On a computer with a machine precision of $2^{-24} \approx 6 \times 10^{-8}$ then $1 + \epsilon^2 > 1$ only for $\epsilon > 2.45 \times 10^{-4}$.

This difficulty is overcome using the QR decomposition or the SVD. It is easy to verify that

$$A^+ = (A^T A)^{-1} A^T = R^{-1} Y^T.$$

Thus the least squares solution to $A\mathbf{x} = \mathbf{b}$ is calculated by solving

$$R\mathbf{x} = Y^T \mathbf{b}$$

by back-substitution. The use of orthogonal matrices ensures that the process is numerically stable.

Example A.8.2 (Over-determined linear system using QR factorization) *Let*

$$A = \begin{bmatrix} 3 & -4 \\ 3 & -4 \\ 0 & 6 \end{bmatrix} \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix}$$

then

$$Q = \begin{bmatrix} \frac{-1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & -1 & 0 \end{bmatrix} \quad \textit{and} \quad R = \begin{bmatrix} -3\sqrt{2} & 4\sqrt{2} \\ 0 & -6 \end{bmatrix}.$$

As A is a 3 by 2 matrix of full rank (its columns are linearly independent) as all the diagonal elements of R are non-zero. Hence Y is the matrix formed by the first two columns of Q and Z is the matrix formed by the last column of Q. It is easily verified that $A = YR$ and $Z^T A = 0$. The least squares solution $\mathbf{x}^*$ to $A\mathbf{x} = \mathbf{b}$ is found by solving $R\mathbf{x} = Y^T\mathbf{b}$ by back-substitution to give

$$\mathbf{x}^* = \begin{bmatrix} \frac{1}{18} \\ \frac{1}{6} \end{bmatrix} \quad \textit{and} \quad \mathbf{r}^* = \mathbf{b} - A\mathbf{x}^* = ZZ^T\mathbf{b} = \begin{bmatrix} \frac{5}{2} \\ -\frac{5}{2} \\ 0 \end{bmatrix}.$$

As $\mathbf{r}^* \neq 0$ the vector $\mathbf{b}$ is not in the range space of A.

In MATLAB this example is

```
A = [3 -4; 3 -4; 0 6]
[m, n] = size(A)       % Over-determined example, m > n
b = [2; -3; 1]
[Q, R] = qr(A)
% The following assumes A has full column rank = n
R = R(1:n,1:n)         % Extract upper triangular matrix R
Y = Q(:,1:n)
Z = Q(:,n+1:m)
x = A \ b
x = R \ (Y'*b)
r = b - A*x
r = Z*(Z'*b)
```

A.8.5 Under-determined Linear Systems

If $A \in \mathbb{R}^{m\times n}$ where $m < n$ then $A\mathbf{x} = \mathbf{b}$ represents an under-determined system of m linear equations in n variables $\mathbf{x}$. If the equations are consistent then there are infinitely many points $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$. If $rank(A) = m$ then $AA^T \in \mathbb{R}^{m\times m}$ is nonsingular, and

$$\mathbf{x}^* = A^T(AA^T)^{-1}\mathbf{b}$$

is one of these points. The point $\mathbf{x}^*$ is in fact the point satisfying $A\mathbf{x} = \mathbf{b}$ that is closest to the origin, in the sense that it is the solution to

$$\begin{array}{ll} Minimize & \|\mathbf{x}\|_2 \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & A\mathbf{x} = \mathbf{b}. \end{array} \tag{A.8.1}$$

Rather than forming AA^T it is numerically more stable to use the QR factorization. As the preceding sections considered QR factorizations of matrices with more rows than columns it is convenient to use the QR factorization of A^T. Then

$$A^T = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Y & Z \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix} = YR,$$

where $Y \in \mathbb{R}^{n\times m}$, $Z \in \mathbb{R}^{n\times(n-m)}$ are orthogonal matrices and $R \in \mathbb{R}^{m\times m}$ is an upper-triangular nonsingular (as A has full rank) matrix. It is easily verified that

$$\mathbf{x}^* = A^T(AA^T)^{-1}\mathbf{b} = YR^{-T}\mathbf{b}$$

so $\mathbf{x}^*$ can be obtained by solving the well-determined system $R^T\mathbf{z} = \mathbf{b}$ for $\mathbf{z} \in \mathbb{R}^m$ by forward substitution and then setting $\mathbf{x}^* = Y\mathbf{z}$.

More generally the problem of finding the closest (using the 2-norm) point satisfying $A\mathbf{x} = \mathbf{b}$ to a point $\mathbf{y} \in \mathbb{R}^n$, that is solving

$$\begin{array}{ll} Minimize & \|\mathbf{x} - \mathbf{y}\|_2 \\ \mathbf{x} \in \mathbb{R}^n & \\ Subject\ to & A\mathbf{x} = \mathbf{b} \end{array}$$

has the solution

$$\mathbf{x}^* = A^T(AA^T)^{-1}\mathbf{b} + (I - A^T(AA^T)^{-1}A)\mathbf{y} = YR^{-T}\mathbf{b} + ZZ^T\mathbf{y}.$$

Note that

$$I - A^T(AA^T)^{-1}A = ZZ^T$$

is the orthogonal projection onto the null space of A.

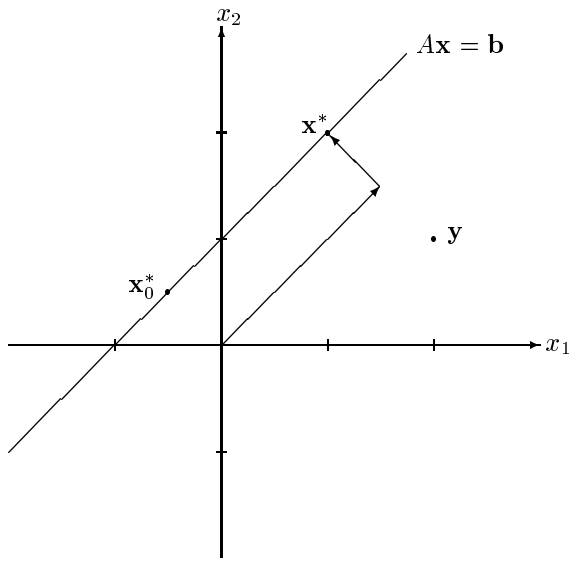


Figure A.8.2: Undetermined linear system, $\mathbf{x}^* = argmin\ \|\mathbf{x} - \mathbf{y}\|_2 : A\mathbf{x} = \mathbf{b}$

Example A.8.3 (Under-determined linear system) *Let* $A = \begin{bmatrix} -1 & 1 \end{bmatrix}$ *and* $\mathbf{b} = 1$ *so* $m = 1$ *and* $n = 2$. *The point satisfying* $A\mathbf{x} = \mathbf{b}$ *which is closest to the origin, so it solves (A.8.1), is*

$$\mathbf{x}_0^* = YR^{-T}\mathbf{b} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

If $\mathbf{y} = \begin{bmatrix} 2 & 1 \end{bmatrix}^T$ *then*

$$\mathbf{x}^* = \mathbf{x}_0^* + ZZ^T\mathbf{y} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} + \begin{bmatrix} \frac{3}{2} \\ \frac{3}{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

is the point satisfying $A\mathbf{x} = \mathbf{b}$ *that is closest to* $\mathbf{y}$ *(see Figure A.8.2). The second term is the projection of* $\mathbf{y}$ *onto the null space of* A*, so is parallel to the line* $A\mathbf{x} = \mathbf{b}$.

For under-determined systems the MATLAB \ operator gives a solution of the linear system. Thus the above example is

```
A = [-1 1]
[m, n] = size(A)    % Under-determined example with m < n
b = 1
x0 = pinv(A)*b      % Least squares solution using pseudo-inverse
x = A \ b           % A solution to A*x = b, but not least squares solution
[Q, R] = qr(A')
R = R(1:m,1:m)      % Assumes A has full rank
Y = Q(:,1:m)
Z = Q(:,m+1:n)
x0 = Y * (R'\b)     % Point satisfying A*x = b closest to the origin
y = [2; 1]
ZZy = Z*(Z'*y)
xs = x0 + ZZy       % Point satisfying A*x = b closest to y
```

A.9 Implementations

The two key issues in most matrix computations are

- The numerical stability of the algorithm - that is how will errors in the data be affected by the calculations and what is the accuracy of the final computed solution.
- The time taken by the algorithm - typically this is estimated by the number of floating point operations required. In particular how does the time taken increase as the size of the problem increases?

A.9.1 Floating point operations

A floating point operation (FLOP) is an addition, subtraction, multiplication or a division in which the arguments are floating point numbers. Counting FLOPS ignores integer arithmetic in calculating vector and array subscripts, and architectures on which division is much more time consuming that multiplication. The computational complexity of common matrix operations is listed in Table A.9.1, where $\alpha, \beta \in \mathbb{R}$ are scalars, $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$, $A, B, C \in \mathbb{R}^{n\times n}$, $L \in \mathbb{R}^{n\times n}$ is lower triangular, $U \in \mathbb{R}^{n\times n}$ is upper-triangular, $P \in \mathbb{R}^{n\times n}$ is a permutation matrix, $X \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{m\times m}$ is orthogonal, and $R \in \mathbb{R}^{n\times n}$ is upper-triangular. In Table A.9.1 only the highest order term is given. Details of each algorithm can be found in

Operation	Example	Complexity
vector–vector multiplication	$\mathbf{z} = \alpha\mathbf{x} + \mathbf{y}$	$2n$
matrix–vector multiplication	$\mathbf{z} = A\mathbf{x} + \beta\mathbf{y}$	$2n^2$
back-substitution	solve $U\mathbf{x} = \mathbf{y}$	n^2
forward-substitution	solve $L\mathbf{y} = \mathbf{b}$	n^2
matrix–matrix multiplication	$C = AB$	$2n^3$
Cholesky factorization	$A = LL^T$	$\frac{n^3}{3}$
LU factorization	$PA = LU$	$\frac{2n^3}{3}$
QR factorization	$X = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$	$2n^2(m - \frac{n}{3})$

Table A.9.1: Complexity of basic matrix operations

Golub and van Loan [GV96] or the LAPACK guide [ABB+95].

The major cost in solving an n by n linear system $A\mathbf{x} = \mathbf{b}$ is the calculation of the LU factorization, which takes $\frac{2n^3}{3} + O(n^2)$ operations. After calculating the LU factorization, the linear system can be solved for several different right-hand-side vectors $\mathbf{b}$ for an additional cost of $O(n^2)$ operations. For large n this is insignificant compared to the cost of calculating the LU factorization.

A.9.2 Memory access

For large problems algorithms with the same number of floating point operations, but different memory access patterns can vary significantly in the their efficiency. This is particularly true of machines with a memory hierarchy, for example workstations with registers on the processor, primary and secondary cache, main memory and virtual memory on disk. As the amount of storage increases the speed of access becomes much slower.

Example A.9.1 *A naive implementation of the matrix–matrix multiplication is*

Algorithm A.9.2 (Row oriented matrix multiplication)

$$
\begin{aligned}
&\textit{for } i = 1, \ldots, n \\
&\quad \textit{for } j = 1, \ldots, n \\
&\qquad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \\
&\quad \textit{endfor} \\
&\textit{endfor}
\end{aligned}
$$

In languages like Fortran which store arrays by columns this algorithm is inefficient as the innermost loop goes across rows, generating large steps through memory and hence more cache misses. For $j = 1, \ldots, n$ let $A_j, B_j, C_j \in \mathbb{R}^n$ denote the jth columns of A, B and C. A column oriented version of matrix multiplication calculates the jth column of C by $C_j = \sum_{k=1}^{n} A_k b_{kj}$, giving

Algorithm A.9.3 (Column oriented matrix multiplication)

```
for j = 1,...,n
    for i = 1,...,n
        c_ij = 0
    endfor
    for k = 1,...,n
        for i = 1,...,n
            c_ij = c_ij + a_ik b_kj
        endfor
    endfor
endfor
```

Algorithm A.9.3 has an innermost loop over the index i which moves down columns of the matrices C and A thus generating sequential memory accesses.

Even more efficient algorithms can be developed by calculating a block *of the matrix C at a time. For simplicity assume both $n = pn_r$ and $n = qn_c$. The blocked algorithm calculates, in a column oriented way, the $n_r \times n_c$ sub-block of the matrix C.*

Algorithm A.9.4 (Blocked matrix multiply)

```
for j_b = 1,...,q
    for i_b = 1,...,p

        for j = (j_b - 1)n_c + 1,...,j_b n_c
            for i = (i_b - 1)n_r + 1,...,i_b n_r
                c_ij = 0
            endfor
            for k = 1,...,n
                for i = (i_b - 1)n_r + 1,...,i_b n_r
                    c_ij = c_ij + a_ik b_kj
                endfor
            endfor
        endfor

    endfor
endfor
```

Each of the Algorithms A.9.3, A.9.3 and A.9.4 perform exactly the same number of floating point operations (multiplications and additions). However the different memory access patterns can produce different calculation times. The times taken for these these algorithms on a Silicon Graphics Challenge computer using 150 MHz R4400 processors with 16 Kbyte of on-chip cache and 1 Mbyte of secondary cache are given in Table A.9.2. The blocked algorithms (which had $n_r = 40$ and $n_c = 25$), and in particular the

Size	Algorithm			
n	Row	Column	Block	DGEMM
200	1.0	1.0	0.8	0.5
300	5.3	3.4	2.7	1.5
400	19.6	10.1	6.5	3.8
500	41.5	24.4	12.8	7.7
512	195.1	27.4	22.6	11.5
600	87.2	42.7	22.5	12.1

Table A.9.2: Matrix multiply times in seconds

level III BLAS routine DGEMM as provided by the manufacturer, are the most efficient for large matrices as they make best use of the cache. Note that a matrix size which is a multiple of the cache line (for example $n = 512$ in Table A.9.2) can dramatically slow down the computation.

Fortunately the level of detail considered in Example A.9.1 is rarely necessary as mathematical libraries with good implementations of the BLAS are available from the computer manufacturer or as part of general purpose mathematical libraries such as NAG [NAG] and Visual Numerics (previously IMSL) [Vis]. The vendors of most workstations also usually provide optimized versions of the BLAS for their particular architectures. Thus using the BLAS for matrix operations provides an efficient portable way of implementing algorithms that are based on matrix operations. The three main levels of the BLAS are outlined in Table A.9.3.

Level	Type of operations	Example	Name
I	vector–vector	$\alpha\mathbf{x} + \mathbf{y}$	DAXPY
II	matrix–vector	$\alpha A\mathbf{x} + \beta\mathbf{y}$	DGEMV
III	matrix–matrix	$\alpha AB + \beta C$	DGEMM

Table A.9.3: BLAS I, II, III

Efficient routines for a wide range of matrix operations are available through the LAPACK project [ABB+95] which uses the level III BLAS. LAPACK is a successor to the widely used LINPACK [DBMS79] and EISPACK [SBD+67, GBDM77] libraries. Matrix languages such as MATLAB include efficient versions of many essential matrix operations.

A.9.3 Vectorization

Many languages and computer architectures are more efficient if operations are presented in terms of vector and matrix operations. This is particularly true of interpreted languages like MATLAB, and Fortran 90 compilers, both of which include many array operations. Using array operations can simplify the coding of the algorithm, and give full scope for vectorization, and possibly even parallelization, of the algorithm. One trade off can be increased storage requirements.

The following example using MATLAB to calculate an integral using the Trapezoidal rule illustrates these trade-offs.

```
% Calculate integral of sin(x) over 0 to pi
% Uses trapezoidal rule which has error O(h^2)
% Exact value of integral is 2

format compact
format long

clear                    % Clear all variables
```

```
n = 10000                 % Number of intervals; Try different values
a = 0
b = pi                    % Mathematical constant pi
h = (b-a) / n             % Width of each interval

% Sequential programming style with explicit loop
tic                       % Start timing

ints = sin(a)/2;
for i = 1:n-1
   z = a + i*h;
   ints = ints + sin(z);
end
ints = ints + sin(b)/2;
ints = h*ints

toc                       % Finish timing
whos                      % List current storage

% Vectorised version using 2 vectors x and f of length n+1
tic                       % Start timing

x = a + h*[0:n];
f = sin(x);
intv = h*(f(1)/2 + sum(f(1:n-1)) + f(n)/2)

toc                       % Finish timing
whos                      % List current storage
```

On a 60 MHz Pentium with $n = 10,000$ the serial loop version took around 2.8 seconds, while the vectorised version took around 0.17 seconds. This represents a speedup of over 17 times. However the serial version only requires the storage of scalars, while the vectorised version uses $2n + O(1)$ storage.

A.10 References

Golub and van Loan [GV96] provides an excellent reference on matrix computations, but at a relatively advanced level. Strang [Str88] provides an easier paced introduction to numerical linear algebra. Other general references on matrix computations include [Ste73]. Goldberg [Gol91] covers the essentials of floating point arithmetic. Higham [Hig96] has lots of material on numerical accuracy issues. The classical reference on eigenvalue problems is Wilkinson [Wil65], while Parlett [Par80] covers the symmetric eigenvalue problem. Least squares problems are considered by Lawson and Hanson [LH95] and the recent book by Bjorck [AB96]. Methods for solving large sparse positive definite linear systems are discussed by George and Liu [GL89]. Gill, Murray and Wright [GMW91] discuss matrix computations in the context of optimization methods.

A good introduction to MATLAB is Pratap [Pra01], while [HH00] contains more details on efficient use of MATLAB. Information about MATLAB is also available from the news group `comp.soft-sys.matlab` and from the world wide web (see below).

The BLAS were started in [LHKK79], and optimized versions are now provided by most manufacturers. Coleman and van Loan [CV88] discuss the BLAS and MATLAB. The LINPACK [DBMS79] and EISPACK [SBD+67, GBDM77] libraries have been largely replaced by the LAPACK [ABB+95] library, which uses the BLAS level III for efficiency of on a wide range of architectures. Many routines are available electronically from NETLIB [DG87]. General mathematical libraries, such as the Numerical Algorithms Group (NAG) library [NAG] and the International Mathematical and Statistical Library (IMSL) [Vis],

include routines for a wide variety of matrix computations, as well as many other tasks. Fortran and C routines for many matrix computations are also available from the *Numerical Recipes* [PTVF92]. Algorithms and software for matrix computations on advanced computer architectures are covered in [DDSV90, DW95, PTVF96].

Information on software is available from the world wide web. Relevant addresses are

```
http://www.mathworks.com                    % Matlab
http://www.netlib.org                       % NETLIB
http://www.netlib.org/lapack/index.html     % LAPACK
http://www.nag.co.uk                        % NAG
http://www.vni.com                          % IMSL
```

Appendix B

Basic Statistics

This appendix summarizes some of the elementary statistics which is important in the application of optimization methods, particularly in areas such as portfolio optimization and stochastic programming.

B.1 Random variables

A *random variable* Y models a quantity whose value is uncertain. Examples include the demand for a product, the annual return on a stock, the amount a stock will move from its current value, and the interest rate over the next 30 days. Random variables may be either *continuous* where Y can take any value in a set $\Omega \subseteq \mathbb{R}$, or *discrete* where Y can only take a finite number m of possible values y_i for $i = 1, \ldots, m$. A continuous random variable Y is described by its *probability density function* (pdf) $\rho(y)$ for $y \in \Omega$, where

$$\rho(y) \geq 0 \quad \forall\, y \in \Omega \quad \text{and} \quad \int_\Omega \rho(y)\, dy = 1.$$

A discrete random variable is described by probabilities p_i for $i = 1, \ldots, m$, where

$$p_i \geq 0 \quad i = 1, \ldots, m \quad \text{and} \quad \sum_{i=1}^{m} p_i = 1.$$

B.1.1 Probabilities

The probability that a random variable takes a value in the interval $(a, b]$ is

$$\mathcal{P}[\, a < y \leq b \,] = \int_a^b \rho(y)\, dy$$

for a continuous distribution, and

$$\mathcal{P}[\, a < y \leq b \,] = \sum_{i=a+1}^{b} p_i$$

for a discrete distribution. The *median* $y_{0.5}$ is determined by the equation

$$\mathcal{P}[\, Y \leq y_{0.5} \,] = \int_{-\infty}^{y_{0.5}} \rho(y)\, dy = 0.5.$$

Similarly the 95th *percentile* is $y_{0.95}$ such that

$$\mathcal{P}[\, Y \leq y_{0.95} \,] = \int_{-\infty}^{y_{0.95}} \rho(y)\, dy = 0.95.$$

These probabilities are closely related to the *cumulative density functions* (cdf)

$$\mathcal{P}[\,Y \leq \bar{y}\,] = \int_{-\infty}^{\bar{y}} \rho(y)dy,$$

which gives the probability that the random variable Y takes a value less than or equal to $\bar{y}$. For a discrete random variable the cumulative density function is

$$\mathcal{P}[\,Y \leq \bar{y}\,] = \sum_{i=1}^{\bar{y}} p_i.$$

B.1.2 Mean and Variance

The *expected value* $\mathcal{E}[\,Y\,]$ of Y (or mean μ) is

$$\mathcal{E}[\,Y\,] \equiv \mu = \sum_{i=1}^{m} p_i y_i.$$

For a continuous random variable Y which can take values in the set Ω and with probability density function $\rho(y)$ the expected value is

$$\mathcal{E}[\,Y\,] \equiv \mu = \int_{\Omega} \rho(y) y \; dy.$$

More generally the expected values of a function $\phi(Y)$ is

$$\mathcal{E}[\,\phi(Y)\,] = \sum_{i=1}^{m} p_i \phi(y_i)$$

for a discrete random variable, or

$$\mathcal{E}[\,\phi(Y)\,] = \int_{\Omega} \rho(y)\phi(y) \; dy.$$

for a continuous random variable. The kth moment of a discrete random variable is

$$M_k = \sum_{i=1}^{m} p_i y_i^k,$$

while the kth moment on a continuous random variable is

$$M_k = \int_{\Omega} \rho(y) y^k \; dy.$$

The *variance* $\mathcal{V}[\,Y\,]$ is defined by

$$\begin{aligned} \mathcal{V}[\,Y\,] &\equiv \mathcal{E}[\,(Y-\mu)^2\,] \\ &= \mathcal{E}[\,Y^2\,] - \mu^2. \end{aligned}$$

The *standard deviation* $\sigma = \sqrt{\mathcal{V}[\,Y\,]}$, and $\mathcal{V}[\,Y\,] \geq 0$. For a discrete random variable Y

$$\begin{aligned} \mathcal{V}[\,Y\,] &= \sum_{i=1}^{m} p_i (y_i - \mu)^2 \\ &= \sum_{i=1}^{m} p_i y_i^2 - \mu^2. \end{aligned}$$

The *skewness* of a continuous distribution is

$$\beta_1 = \mathcal{E}[\,(x-\mu)^3\,]/\sigma^2,$$

while the *kurtosis* is

$$\beta_2 = \mathcal{E}[\,(x-\mu)^4\,]/\sigma^4.$$

Skewness is positive for distributions with more weight above the mean, and negative for distributions with more weight below the mean. Distributions with large kurtosis are those with fat tails.

B.1.3 Normal and Lognormal distributions

Common distributions are the Normal distribution and the Log-normal distribution (see [AB69, EMH+93] for example). The normal distribution is completely characterized by its mean μ and variance σ^2 (standard deviation σ). The notation $Y \sim N(\mu, \sigma^2)$ is used to describe a random variable Y which is normally distributed with mean μ and variance σ^2. The probability density function for $Y \sim N(\mu, \sigma^2)$ is

$$\rho(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}.$$

The first two plots in Figure B.1.1 illustrate the continuous probability density function (pdf) and the

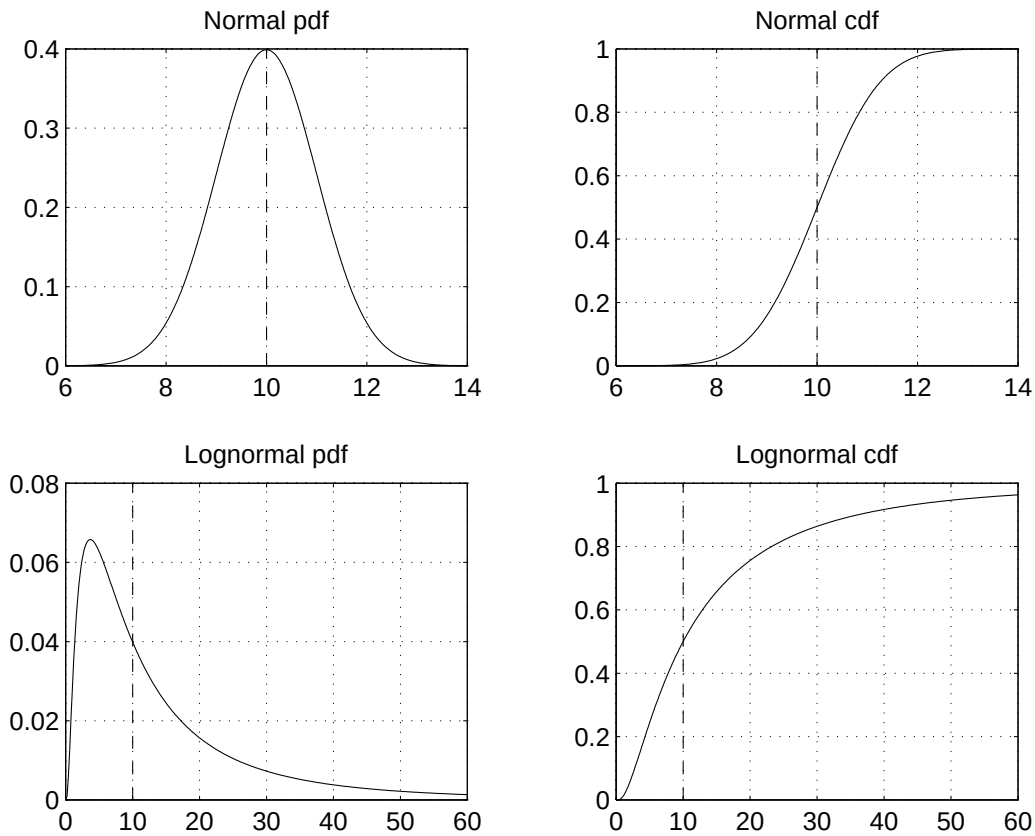


Figure B.1.1: Continuous Normal and Log-normal distributions

cumulative density function (cdf) for a normal distribution (mean $\mu = 10$, standard deviation $\sigma = 1$). For a normally distributed random variable

1. 68.26% of the values are within 1 standard deviation of the mean.
2. 95.46% of the values are within 2 standard deviations of the mean.
3. 99.74% of the values are within 3 standard deviations of the mean.
4. Effectively 100% of the values are within ± 5 standard deviations of the mean.

If $\log Y$ is normally distributed, then Y has a log-normal distribution. If $\log Y \sim N(\mu, \sigma^2)$ then $\mathcal{E}[\,Y\,] = e^{\mu + \frac{1}{2\sigma^2}}$ and $\exp Y^2 = e^{2\mu + 2\sigma^2}$ so the variance is $\mathcal{V}[\,Y\,] = e^{2\mu+2\sigma^2}(e^{\sigma^2} - 1)$. The second example in Figure B.1.1 is a Lognormal distribution with mean $\mathcal{E}[\,Y\,] = \log 10$ and standard deviation $\sigma = 1$.

The Lognormal distribution is used to model the distribution of a random variable which only takes non-negative values and does not have a symmetric distribution. For example

$$\text{Stock return} = \frac{\text{Final price of stock} + \text{Dividends}}{\text{Initial price of stock}}.$$

From this definition the stock return must be nonnegative, and this fact is reflected in the log-normal distribution. Another quantity that is frequently assumed to be log-normally distributed is the (risk free) interest rate which is always nonnegative. See [AB69] and [MGB74, p. 540-541].

Figure B.1.2 illustrates discrete probability density functions and discrete cumulative density functions with $m = 20$ points.

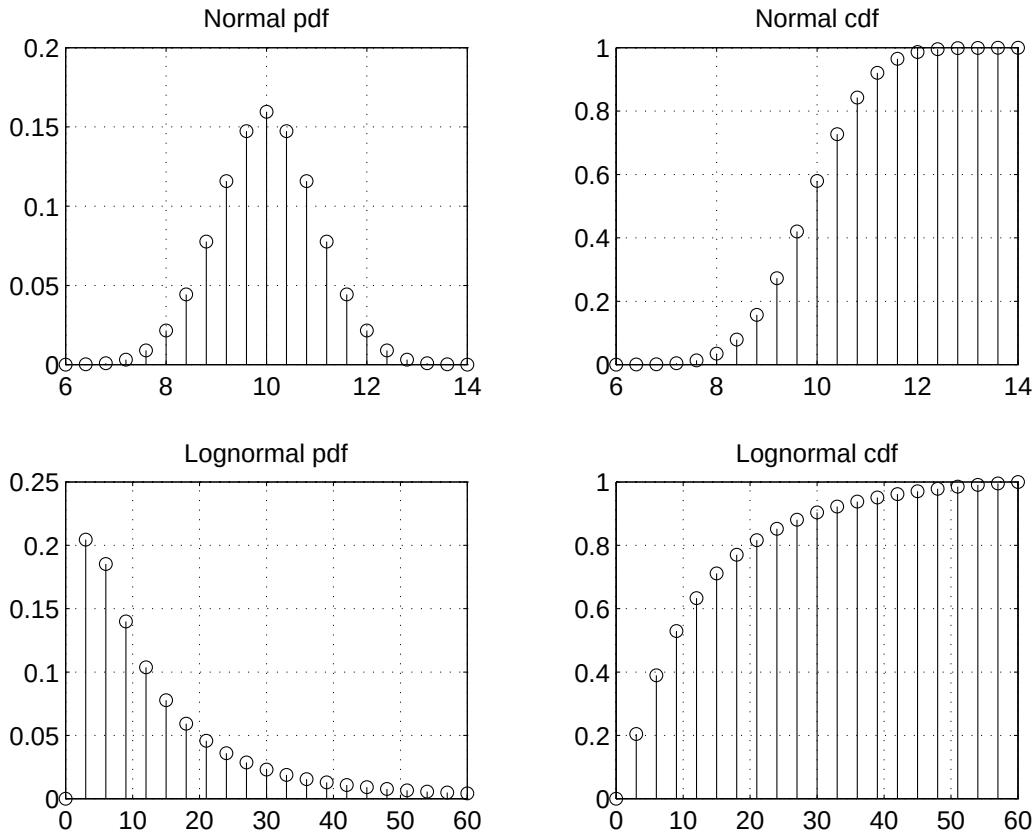


Figure B.1.2: Discrete Normal and Log-normal distributions

The error function

$$\operatorname{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y e^{-t^2}\, dt$$

is closely related to the standard normal cumulative density fucntion $\Phi(y)$ by

$$\Phi(y) = \tfrac{1}{2}(1 + \operatorname{erf}(y/\sqrt{2})).$$

Another important function is the inverse standard normal cumulative density function $\Phi^{-1} : [0,1] \to \mathbb{R}$ defined by

$$\Phi^{-1}(p) = y \iff \Phi(y) = p \quad p \in [0,1].$$

B.1.4 Correlation

When there is an interaction between two or more random variables the *covariance* measures the degree to which two random variables move together. For example if the extreme positive values for one random

variable are associated with extreme negative values for the other random variable, and the random variables move in opposite directions, they have a negative covariance. The values of random variables with positive and negative correlation are plotted in Figure B.1.3.

Let Y be a random variable with mean μ and Z be a random variable with mean ν. The *covariance* of Y and Z is

$$Cov[Y,Z] \equiv \mathcal{E}[\,(Y-\mathcal{E}[Y])(Z-\mathcal{E}[Z])\,] = \mathcal{E}[\,(Y-\mu)(X-\nu)\,].$$

Note that

$$\begin{aligned} Cov[Y,Y] &= \mathcal{V}[Y], \\ Cov[Y,Z] &= \mathcal{E}[YZ]-\mathcal{E}[Y]\mathcal{E}[Z]. \end{aligned}$$

The *correlation* between two random variables Y and Z is

$$Cor[Y,Z] = \frac{Cov[Y,Z]}{\sqrt{\mathcal{V}[Y]\mathcal{V}[Z]}} \quad \in [-1,1].$$

Let Y be a discrete random variable having value y_i with probability p_i for $i=1,\ldots,m$, and let Z have value z_j with probability q_j for $j=1,\ldots,k$. Then the *covariance* is

$$\begin{aligned} Cov[Y,Z] &= \mathcal{E}[\,(Y-\mu)(Z-\nu)\,] \\ &= \sum_{i=1}^{m}\sum_{j=1}^{k} p_i q_j (y_i-\mu)(z_j-\nu) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{k} p_i q_j y_i z_j - \mu\nu. \end{aligned}$$

Let $\alpha,\beta \in \mathbb{R}$, and let $W=\alpha Y+\beta Z$. Some basic properties of linear combinations of random variables are:

$$\begin{aligned} \mathcal{E}[W] &= \alpha\mathcal{E}[Y]+\beta\mathcal{E}[Z] = \alpha\mu+\beta\nu \\ \mathcal{V}[W] &= \mathcal{E}[\,(W-\mathcal{E}[W])^2\,] \\ &= \mathcal{E}[\,(\alpha Y+\beta Z-\alpha\mu-\beta\nu)^2\,] \\ &= \alpha^2\mathcal{V}[Y]+\beta^2\mathcal{V}[Z]+2\alpha\beta Cov[Y,Z]. \end{aligned}$$

The variance of the sum of two random variables is the sum of the variances of each random variable plus twice the covariance of the random variables. *Thus by choosing variables with a negative covariance it is possible to get the variance of the sum of the two random variables less than the sum if the individual variances.* This is the basis of portfolio optimization considered in Chapter 8.

Example B.1.1 (Covariance) *Consider three random variables Y_1, Y_2, Y_3 which can take the $m=12$ values given in Table B.1.1 with equal probability $p_i = 1/m$ (i.e. they are uniformly distributed). The*

	1	2	3	4	5	6	7	8	9	10	11	12
Y_1	5	15	20	10	0	-10	-20	0	8	16	4	-12
Y_2	3	12	10	8	4	-5	-10	-5	4	12	5	-2
Y_3	0	-10	-5	8	10	15	12	5	-8	-6	0	15

Table B.1.1: Value of uniformly distributed random variables

values in Table B.1.1 are plotted in Figure B.1.3. The mean of each of these random variables is 3, so the mean vector is $\mu = [\,3 \quad 3 \quad 3\,]^T$. The covariance matrix for these three random variables is

$$C = \begin{bmatrix} 147.4545 & 80.4545 & -91.6364 \\ 80.4545 & 51.2727 & -48.0000 \\ -91.6364 & -48.0000 & 81.8182 \end{bmatrix},$$

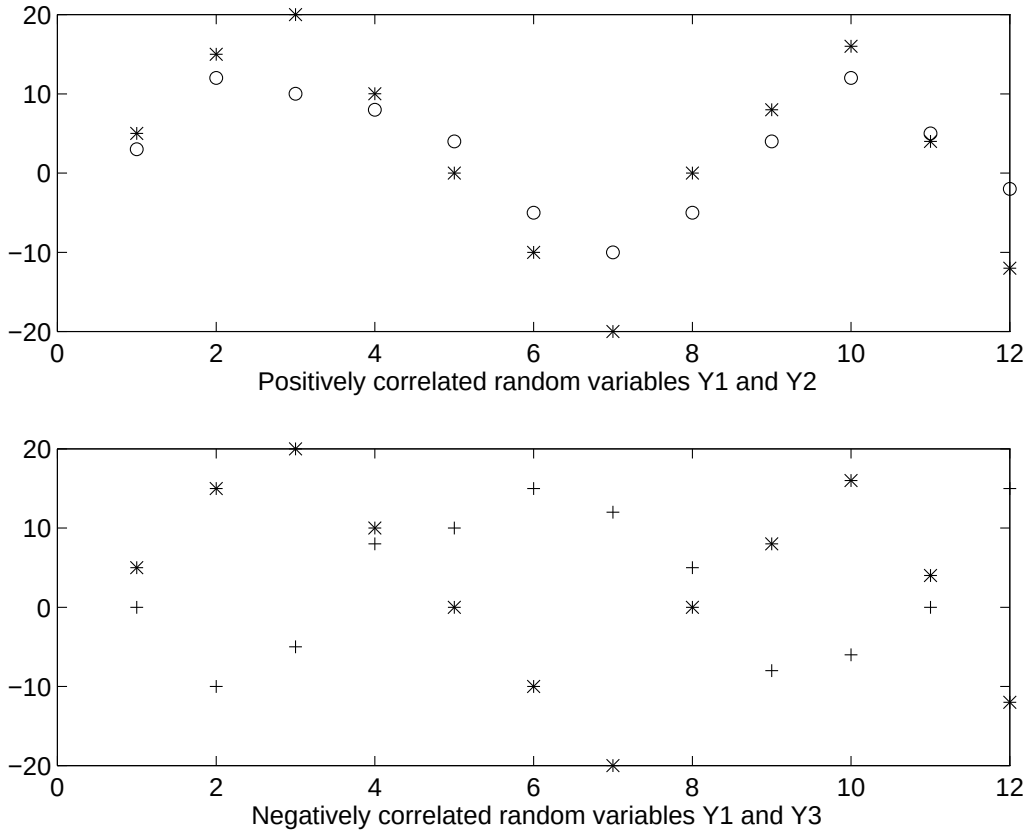


Figure B.1.3: Positive and negative correlations

while the correlation matrix is

$$K = \begin{bmatrix} 1 & 0.9253 & -0.8343 \\ 0.9253 & 1 & -0.7411 \\ -0.8343 & -0.7411 & 1 \end{bmatrix}.$$

The covariance matrix C can easily be calculated by the MATLAB *commands*

```
C = cov(Y)
```

and and the correlation coefficients K by the command

```
K = corrcoef(Y)
```

where Y is a m by 3 matrix whose columns contain the rows in Table B.1.1. Thus the random variables Y_1 and Y_2 are positively correlated, while the random variables Y_1 and Y_3 are negatively correlated.

B.2 Random vectors

Frequently there are several quantities which are not independent of each other, so the joint distribution must be considered. For example if the random variable Y_1 may represents the interest rate and Y_2 represents the return on a particular stock, then changes in interest rates typically affect stock prices, so the two random variables are not independent.

Consider random variables $Y_1, \ldots, Y_n$ which form a random vector $\mathbf{Y} \in \mathbb{R}^n$. The means $\mu_i = \mathcal{E}[\,Y_i\,]$ form a vector $\mu \in \mathbb{R}^n$, and the covariance matrix

$$C_{ij} = \mathcal{E}[\,(Y_i - \mu_i)(Y_j - \mu_j)\,] \quad i, j = 1, \ldots, n$$

is an n by n symmetric, positive semidefinite matrix. The correlation matrix is

$$K = D^{-1} C D^{-1}$$

where $D = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, so $D^2 = \mathrm{diag}(C)$.

In fact a symmetric matrix is a covariance matrix if and only if it is positive semidefinite (see [Mui82] for example).

B.2.1 Multivariate normal distribution

The multivariate normal density function is completely characterized by the vector of means $\mu \in \mathbb{R}^n$ and the covariance matrix C. For a nonsingular covariance matrix it has density function

$$\phi(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det C}}\, e^{-(\mathbf{x}-\mu)^T C^{-1}(\mathbf{x}-\mu)/2} \quad \mathbf{x} \in \mathbb{R}^n. \tag{B.2.1}$$

Calculating a probability for a multivariate normal distribution is the (for $n > 2$ challenging) problem of numerically evaluating the multiple integrals

$$\mathcal{P}[Y \le \mathbf{a}] = \int_{-\infty}^{a_1} \cdots \int_{-\infty}^{a_n} \phi(\mathbf{x}) dx_n \ldots dx_1 \tag{B.2.2}$$

$$\mathcal{P}[\mathbf{a} \le Y \le \mathbf{b}] = \int_{a_1}^{b_1} \cdots \int_{a_n}^{b_n} \phi(\mathbf{x}) dx_n \ldots dx_1 \tag{B.2.3}$$

$$\mathcal{P}[Y \ge \mathbf{b}] = \int_{b_1}^{\infty} \cdots \int_{b_n}^{\infty} \phi(\mathbf{x}) dx_n \ldots dx_1. \tag{B.2.4}$$

The expected value of a function $f : \mathbb{R}^n \to \mathbb{R}^n$ of a multivariate random variable is

$$\mathcal{E}[f] = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(\mathbf{x})\phi(\mathbf{x}) dx_n \ldots dx_1.$$

Bivariate normal

In particular the *bivariate normal distribution* has

$$C = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \Longrightarrow C^{-1} = \frac{1}{\sigma_1^2\sigma_2^2(1-\rho^2)} \begin{bmatrix} \sigma_2^2 & -\rho\sigma_1\sigma_2 \\ -\rho\sigma_1\sigma_2 & \sigma_1^2 \end{bmatrix}$$

where the variances $\sigma_1^2 \neq 0$, $\sigma_2^2 \neq 0$ and the correlation $\rho \neq \pm 1$. The bivariate normal density function is then

$$\phi(x) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}\, e^{-\dfrac{\psi(x)}{2(1-\rho^2)}},$$

where

$$\psi(x) = \left(\frac{x_1-\mu_1}{\sigma_1}\right)^2 - 2\rho\left(\frac{x_1-\mu_1}{\sigma_1}\right)\left(\frac{x_2-\mu_2}{\sigma_2}\right) + \left(\frac{x_2-\mu_2}{\sigma_2}\right)^2.$$

Bibliography

[AB69] J. Aitchinson and J. A. C. Brown. *The Lognormal Distribution with Special Reference to its Uses in Economics.* Cambridge University Press, 1969.

[AB96] Ake Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[ABB+95] E. Anderson, Z. Bai, C. Bischoff, James Demmel, Jack J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and Danny C. Sorensen. *LAPACK Users' Guide.* SIAM, Philadelphia, second edition, 1995.

[Arm66] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[Bai88] David H. Bailey. Extra high speed matrix multiplication on the Cray-2. *SIAM Journal on Scientific and Statistical Computing*, 9, 1988.

[BKM88] Anthony Brooke, David Kendrick, and Alexander Meerhaus. *GAMS: A User's Guide.* The Scientific Press, 1988.

[BKM96] Zvi Bodie, Alex Kane, and Alan J. Marcus. *Investments.* Irwin, Chicago, 3rd edition, 1996.

[BL97] John R. Birge and Francois Louveaux. *Introduction to Stochastic Programming.* Springer-Verlag, New York, 1997.

[BLS91] David H. Bailey, King Lee, and Horst D. Simon. Using Strassen's algorithm to accelerate the solution of linear systems. *Journal of Supercomputing*, 4:357–371, 1991.

[BM82] Johannes Bisschop and Alexander Meerhaus. On the development of a general algebraic modelling system in a strategic planning environment. *Mathematical Programming*, 20:1 – 29, 1982. Introduction to GAMS.

[BP94] Dario Bini and Victor Y. Pan. *Polynomial and Matrix Computations. Volume 1: Fundamental Algorithms.* Birkhäuser, Boston, 1994.

[Bre73] Richard P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Bro70] Charles G. Broyden. The convergence of a class of double rank minimization algorithms: Parts I and II. *Journal of the Institute of Mathematics and its Applications*, 6:76–90 and 222–231, 1970.

[CCPS98] W. J. Cook, W.H. Cunningham, W.R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization.* Addison-Wesley, Reading, MA, 1998.

[Cha91] Bruce W. Char. *Maple V language reference manual.* Springer-Verlag, Berlin and New York, 1991.

[Cha92] Bruce W. Char. *First leaves : a tutorial introduction to Maple V.* Springer-Verlag, Berlin and New York, 1992.

[Chv83] Vasek Chvátal. *Linear Programming.* W. H. Freeman, 1983.

[Cla83] F. H. Clarke. *Optimization and Nonsmooth Anlysis.* John Wiley, Chichester and New York, 1983.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Transactions on Mathematical Software*, 13(3):272–280, 1987.

[Col84] Thomas F. Coleman. *Large Sparse Numerical Optimization*, volume 165 of *Lecture Notes in Computer Science.* Springer-Verlag, 1984.

[Cor95] Robert M. Corless. *Essential Maple: An Introduction for Scientific Programmers.* Springer-Verlag, Springer-Verlag, 1995.

[CV88] Thomas F. Coleman and Charles F. Van Loan. *Handbook for Matrix Computations.* SIAM, Philadelphia, 1988.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 1–6. ACM Press, New York, 1987.

[Dan59] George B. Dantzig. *Linear Programming and Extensions.* Rand Corporation, 1959.

[Dan91] George B. Dantzig. *Linear Programming: The story about how it began*, pages 19 – 31. In Lenstra et al. [LRS91], 1991.

[Dav59] William C. Davidon. Variable metric methods for minimization. Technical Report ANL-5990, Argonne National Laboratory, 1959.

[Dav91] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[DBMS79] Jack J. Dongarra, James R. Bunch, Cleve B. Moler, and G. W. Stewart. *LINPACK Users' Guide.* SIAM, Philadelphia, 1979.

[DDSV90] Jack J. Dongarra, Ian S. Duff, Danny C. Sorensen, and Henk A. Van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, Philadelphia, 1990.

[DES82] Ron S. Dembo, S. C. Eisenstat, and Trond Steinhaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.

[DG87] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Communications of the ACM*, 30:403–407, 1987.

[DH92] James Demmel and Nicholas J. Higham. Stability of block algorithms with fast Level 3 BLAS. *ACM Transactions on Mathematical Software*, 18(3):274–291, 1992.

[Dix72] Lawrence C. W. Dixon. Quasi-newton algorithms generate identical points. *Mathematical Programming*, 2:383–387, 1972.

[DM74] V. F. Demyanov and V. N. Molozemov. *Introduction to Minimiax.* John Wiley, Chichester and New York, 1974.

[DM77] John E. Dennis and Jorge J. Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977.

[DS83] John E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Least Squares.* Prentice Hall, Englewood Cliffs, NJ, 1983.

[DW95] Jack J. Dongarra and David W. Walker. Software libraries for linear algebra computations on high performance computers. *SIAM Review*, 37(2):151–180, June 1995.

[EMH+93] Evans, Merran, Hastings, Nicholas, Peacock, and Brian. *Statistical Distributions.* John Wiley, second edition, 1993.

[FGK93] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modelling Language for Mathematical Programming.* The Scientific Press, 1993.

[Fia83] Anthony V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Academic Press, New York and London, 1983.

[Fle70] Roger Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

[Fle87] Roger Fletcher. *Practical Methods of Optimization.* John Wiley, Chichester and New York, second edition, 1987.

[FP63] Roger Fletcher and Michael J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963.

[FR64] Roger Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[FT93] C. D. Feinsten and M. N. Thapa. A reformulation of a mean absolute deviation portfolio optimization model. *Management Science*, 39:1552–1553, 1993.

[GBDM77] B. S. Garbow, J. M. Boyle, Jack J. Dongarra, and Cleve B. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension.* Number 51 in Lecture Notes in Computer Science. Springer-Verlag, Berlin and New York, 1977.

[GC91] Andreas Griewank and George F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation and Application.* SIAM, Philadelphia, 1991.

[GH96] Saul I. Gass and Carl M. Harris. *Encyclopedia of Operations Research and Management Science.* Kulwer, 1996.

[GH97] Walter Gander and Jiri Hrebicek. *Solving problems in Scientific Computing using MAPLE and MATLAB.* Springer-Verlag, Berlin and New York, 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[GL81] Alan George and Joesph W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[GL89] Alan George and Joesph W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1 – 19, 1989.

[GMW81] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic Press, New York and London, 1981.

[GMW91] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization.* Addison-Wesley, Reading, MA, 1991.

[Gol70] Donald Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computing*, 24:23–26, 1970.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Gol91] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[Gro96] Ignacio E. Grossmann. *Global Optimization in Engineering Design.* Kluwer Academic Publishers, Dordrecht, Boston, 1996.

[GV96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore and London, third edition, 1996.

[Han92] Eldon R. Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker, New York, 1992.

[Har91] W. V. Harlow. Asset allocation in a downside–risk framework. *Financial Analysts Journal*, September–October:28–40, 1991.

[Hau97] R. Haugen. *Modern Investment Theory.* Prentice Hall, Englewood Cliffs, NJ, 4th edition, 1997.

[HH00] Desmond J. Higham and Nicholas J. Higham. *MATLAB Guide.* SIAM, Philadelphia, 2000.

[Hig96] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 1996.

[HL90] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, 1990.

[HL93a] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms I: Fundamentals.* Springer-Verlag, Berlin and New York, 1993.

[HL93b] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms II: Advanced theory and bubdle methods.* Springer-Verlag, Berlin and New York, 1993.

[Hol75] J. H. Holland. *Adaption in Natural and Artificial Systems.* University of Michigan Press, 1975. Reprinted by MIT Press, Cambridge MA, 1992.

[HP95] Reiner Horst and Panos M. Pardalos, editors. *Handbook of Global Optimization.* Kluwer Academic Publishers, 1995.

[HPT95] Reiner Horst, Panos M. Pardalos, and Nguyen Van Thoai, editors. *Introduction to Global Optimization.* Kluwer Academic Publishers, 1995.

[HS52] M. R. Hestenes and E. L. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, Section B, 49:408–436, 1952.

[HS96] Julia L. Higle and Suvrajeet Sen. *Stochasitc Decomposition: A statistical method for large scale stochastic linear progrmming.* Kluwer Academic Publishers, Dordrecht, Boston, 1996.

[HT90] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches.* Springer-Verlag, Berlin and New York, 1990.

[Hub77] Peter J. Huber. *Robust Statistical Procedures*, volume 27 of *SIAM Regional Conference Series in Applied Mathematics.* SIAM, 1977.

[Hub81] Peter J. Huber. *Robust Statistics.* John Wiley, Chichester and New York, 1981.

[Ing89] Lester Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8):967–973, 1989.

[IR92] Lester Ingber and Bruce Rosen. Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical and Computer Modelling*, 16(11):87–100, 1992.

[Joh93] Eugene W. Johnson. *Linear Algebra with Maple V.* Brooks/Cole, 1993.

[Kan82] A. Kane. Skewness preference and portfolio choice. *Journal of Financial and Quantitative Analysis*, XVII:15–25, 1982.

[Kar84] Narrendrar Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Kel95] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, Philadelphia, 1995.

[KGV83] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Kin93] Alan J. King. Asymmetric risk measures and tracking models for portfolio optimization under uncertainty. *Annals of Operations Research*, 45:165–178, 1993.

[Kiw85] C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization.* Number 1133 in Lecture Notes in Mathematics. Springer-Verlag, Berlin and New York, 1985.

[KLM84] Y. Kroll, H. Levy, and H. M. Markowitz. Mean–variance versus direct utility maximization. *The Journal of Finance*, 34:47–61, 1984.

[KSY93] H. Konno, H. Shirakawa, and H. Yamazaki. A mean–absolute deviation–skewness portfolio optimization model. *Annals of Operations Research*, 45:205–220, 1993.

[KW94] Peter Kall and Stein W. Wallace. *Stochastic Programming.* John Wiley, Chichester and New York, 1994.

[KY91] H. Konno and H. Yamazaki. Mean–absolute deviation portfolio optimization model and its application to tokyo stock market. *Management Science*, 37:519–531, 1991.

[LH89] M. L. Leibowitz and R. D. Henriksson. Portfolio optimization with shortfall constraints: A confidence limit approach to managing downside risk. *Financial Analysts Journal*, March–April:34–41, 1989.

[LH95] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* SIAM, Philadelphia, 1995. Reprint of 1964 book published by Prentice-Hall.

[LHKK79] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3):308–323, 1979.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Travelling Salesman Problem: A guided tour of combinatorial optimization.* John Wiley, Chichester and New York, 1985.

[LPS92] Julian Laderman, Victor Y. Pan, and Xuan-He Sha. On practical algorithms for accelerated matrix multiplication. *Linear Algebra and its Applications*, 162-164:557–588, 1992.

[LRS91] Jan Karel Lenstra, Alexander H. H. Rinnooy Kan, and Alexander Schrijver, editors. *History of Mathematical Programming: A Collection of Personal Reminiscences.* NorthHolland, Amsterdam, 1991.

[Lue73] David G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA, 1973.

[Lue97] David G. Luenberger. *Investment Science.* Oxford University Press, New York and Oxford, 1997.

[Mar52] Harry M. Markowitz. Portfolio selection. *Journal of FInance*, 7:77–91, 1952.

[Mar59] Harry M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments.* John Wiley, 1959.

[Mar87] Harry M. Markowitz. *Mean Variance Analysis in Portfolio Choice and Capital Markets.* Blackwell, 1987.

[Mat01] MathWorks, Natick, MA, USA. MATLAB *12 Windows Student Version*, 2001.

[Mer71] Robert C. Merton. Optimum consumption and portfolio rules in a continuous time model. *Journal of Economic Theory*, 3:373–413, 1971.

[MGB74] Alexander M. Mood, Franklin A. Graybill, and Duane C. Boes. *Introduction to the Theory of Statistics*. McGraw Hill, third edition, 1974.

[Mic94] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin and New York, second edition, 1994.

[MT94] Jorge J. Moré and David A. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, 20(3):286–307, September 1994.

[Mui82] Robb J. Muirhead. *Aspects of Multivariate Statistical Theory*. John Wiley, Chichester and New York, 1982.

[Mur81] Bruce A. Murtagh. *Advanced Linear Programming: Computation and practice*. McGraw-Hill, 1981.

[MW93] Jorge J. Moré and Stephen Wright. *Optimization Software Guide*. SIAM, Philadelphia, 1993.

[NAG] *NAG numerical Libraries*. Numerical Algorithm Group, Oxford, U.K. http://www.nag.co.uk/numeric.html.

[Nas90] Stephen G. Nash. *A History of Scientific Computing*. ACM Press, New York, 1990.

[Nie92] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, 1992.

[NRKT89] G. L. Nemhauser, A. H. G. Rinnooy-Kan, and M. J. Todd, editors. *Handbooks in Operations Research and Management Science – Optimization*. North-Holland, Amsterdam, 1989.

[NW88] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley, 1988.

[NW99] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[OR70] James M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York and London, 1970.

[Osb85] Michael R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. John Wiley, Chichester and New York, 1985.

[Pan84a] Victor Y. Pan. How can we speed up matrix multiplication. *SIAM Review*, 26:393–415, 1984.

[Pan84b] Victor Y. Pan. *How to Multiply Matrices Faster*. Springer-Verlag, Berlin and New York, 1984.

[Par80] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, 1980.

[PKS96] David J. Pannell, Ross S. Kingwell, and Steven Schilizzi. Debugging mathematical programming models: Principles and practical strategies. *Review of Marketing and Agricultural Economics*, 64(1):86–100, 1996.

[Pol87] Elijah Polak. On the mathematical foundations of nondifferentiable optimization in engineering design. *SIAM Review*, 29:21–89, 1987.

[Pol97] Elijah Polak. *Optimization: Algorithms and Consistent Approximations*. Springer-Verlag, Berlin and New York, 1997.

[Pow64] Michael J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7:155–152, 1964.

[PR69] Elijah Polak and G. Ribiere. Note sur la convergence de methods de directions conjugres. *Revue Francaise Informatione Recherche Operationnelle*, 16:35–43, 1969.

[Pra01] Rudra Pratap. *Getting started with MATLAB Version 6: a quick introduction for scientists and engineers.* Oxford University Press, 2001.

[Pre95] André PreKopa. *Stochastic Programming.* Kluwer Academic Publishers, 1995.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, second edition, 1992. Second edition is also available in C.

[PTVF96] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing.* Cambridge University Press, Cambridge, UK, 1996.

[Red93] Darren Redfern. *The Maple Handbook.* Springer-Verlag, Berlin and New York, 1993.

[Rei94] Frank K. Reilly. *Investment Analysis and Portfolio Management.* Dryden Press, Fort Worth, Tex, 4th edition, 1994.

[Roc70] R. Tyrrel Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, NJ, 1970.

[RT89] Alexander H. H. Rinnooy Kan and G. T. Timmer. *Global Optimization*, pages 631–659. In Nemhauser et al. [NRKT89], 1989.

[Rud88] Andrew Rudd. *Modern Portfolio Theory: the principles of investment management.* Andrew Rudd Publishing, Orinda, Calif, 1988.

[Sai95] Romesh Saigal. *Linear Programming: A Modern Integrated Analysis.* KLuwer Academic Publishers, Boston. MA, 1995.

[SBD+67] B. T. Smith, J. M. Boyle, Jack J. Dongarra, B. S. Garbow, Y. Ikebe, and Cleve B. Moler. *Matrix Eigensystem Routines – EISPACK Guide.* Number 6 in Lecture Notes in Computer Science. Springer-Verlag, Berlin and New York, 1967. 2nd edition 1976.

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley, New Tork and Chichester, 1998.

[SD01] Kermit Sigmon and Timothy A. Davis. *MATLAB Primer.* Chapman & Hall, 6th edition, 2001.

[Sha70a] David F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computing*, 24:647–656, 1970.

[Sha70b] William F. Sharpe. *Portfolio Theory and Capital Markets.* McGraw-Hill, New York, 1970.

[Ste73] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York and London, 1973.

[Str69] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[Str88] Gilbert Strang. *Linear Algebra and its Applications.* Harcourt Brace Jovanovich, San Diego, 3 edition, 1988.

[Tah92] Hamdy A. Taha. *Operations Research: An Introduction.* Prentice Hall, fifth edition, 1992.

[TWW88] J. F. Traub, G .W. Wasilkowski, and H. Wozniakowski. *Information-based Complexity.* Academic Press, Boston, MA, 1988.

[Vis] *The IMSL Fortran Numerical Libraries.* Visual Numerics Inc, Sugar Land, Texas. http://www.vni.com/products/imsl/.

[Wet96] Roger J.-B. Wets. Challenges in stochastic programming. *Mathematical Programming*, 75(2):115–135, November 1996.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.

[Wou79] Arthur Wouk. *A Course of Applied Functional Analysis.* John Wiley, Chichester and New York, 1979.

[Wri97] Stephen Wright. *Primal-dual interior-point methods.* SIAM, Philadelphia, 1997.

[Zen93] Stavros A. Zenios. *Financial Optimization.* Cambridge University Press, Cambridge, UK, 1993.

Index

Active constraints, 83
Affine function, 17
Affine invariance, 48
Affine set, 54
AMPL, 3
Armijo line search, 111
Automatic differentiation, 40

Back substitution, 241
Bandwidth, 226
Basis, 230
BFGS method, 127
Bisection, 102
Bivariate normal, 259
BLAS, 249
Bracketing methods, 101
Bunch-Kaufman factorization, 121, 124

Catastrophic cancellation, 38, 41, 101
Cauchy-Schwarz inequality, 22, 234
Cholesky factorization, 21, 28, 237
Comparing methods, 48
Computational complexity, 41
Computer algebra, 40
Concave programming problem, 60
Condition number, 242
Conjugate, 231
Conjugate directions, 128
Conjugate gradient methods, 142
    linear systems, 143
    preconditioning, 144
Constrained saddle point, 74
Constrained stationary point
    equality constraints, 74
    inequality constraints, 86
Constraints
    discrete, 31
    equality, 4
    inequality, 4
    linear, 31
    network, 31
    none, 31
    nonlinear, 31
    quadratic, 31
    semi-infinite, 31
    simple bounds, 31
Contour, 16
Contrapositive, 26
Convergence conditions, 47, 113
Converse, 26
Convex combination, 52
Convex composite function, 209
Convex hull, 52
Convex programming problem, 60
Correlation, 257
    matrix, 179
Correlation matrix, 259
Covariance, 257
    matrix, 179
Covariance matrix, 258
Critical points, 103
Cubic interpolation, 100
Cumulative density function, 254
Curvature, 18

Decision variables, 4
Degenerate point, 86
Descent direction, 45, 107
Descent method, 45, 107
Determinant, 20, 229, 232
DFP method, 127
Diagonal matrix, 226
Diagonally dominant, 228
Difference approximation
    central, 40
    forward, 40
Direction derivative, 204
Distribution
    bivariate normal, 259
    log-normal, 255
    multivariate normal, 259
    normal, 255
Dot product, 225

Efficient frontier, 177
Eigenvalues, 232
    positive, 20
Eigenvectors, 232
Eigenvlaues
    interleaving, 233
EISPACK, 223
Epigraph, 57

Equations
over-determined, 130
under-determined, 130
well-determined, 130
Error
absolute, 23, 113, 235
relative, 113, 235
Error function, 256
Exact penalty functions, 170
Existence, 14
Expected value, 254
Extrema, 13
existence, 14
global, 12
local, 12

Feasible region, 4, 31
empty, 32
Fill-in, 240
Fixed point iteration, 98
Fletcher – Reeves method, 143
Floating point accuracy, 223
Floating point numbers, 38
Formulating optimization problems, 3
Forward substitution, 241
Fractals, 134
Freudenstein and Roth's equations, 131
Frobenius norm, 236
Function
affine, 17, 29
nonlinear, 30
nonsmooth, 39
partially separable, 30
quadratic, 17, 29
separable, 30
sum of squares, 30

GAMS, 3
Gauss-Newton method, 133
Gaussian elimination, 237
Generalized gradients, 210
Generalized inverse, 243
Genetic algorithms, 13, 219
Global convergence, 45, 114
Global minimizer, 12
Global optimization, 13
Golden Section search, 102
Gradient, 15

Half spaces, 226
Hessian, 16
Hilbert matrix, 242
Huber's M-estimator, 56
Hyperplane, 226

IEEE arithmetic, 38
IMSL, 249
Indefinite, 20, 228
Infimum, 15
Infinity `Inf`, 223
Inner product, 225
Interval analysis, 13
Inverse, 229
Inverse cumulative normal density, 256

Jacobian, 30, 132

Kernel, 230
KKT conditions, 85
Kurtosis, 254

Lagrangian function, 72
LAPACK, 223
Least squares
linear, 243
Levenberg-Marquadt method, 133
Line in $\mathbb{R}^n$, 17, 225
Line search
approximate, 109
Armijo, 111
Line search methods, 45, 108
Linear convergence, 46
Linear system
over determined, 243
under determined, 245
well determined, 241
Linearly independent, 230
LINPACK, 223
Local minimizers, 13
Locally Lipschitz, 103, 210
Log-normal distribution, 255
Logic
contrapositive, 26
converse, 26
necessary and sufficient, 26
necessary condition, 26
sufficient condition, 26
Lower triangular matrix, 226
LU factorization, 237

M-estimator, 56
Machine precision, 38, 223
Maple, 40
Mathematica, 40
Mathematical models, 2
Matlab, 223
Matlab commands
`\`, 241, 244–246
`.*`, 225
`.^`, 234

%, 224
abs, 232, 234
chol, 238
condest, 243
cond, 242, 243
det, 232
diag, 227
eig, 232
eps, 223
eye, 228
hilb, 243
linprog, 9
lu, 237
max, 234
nnz, 227
norm, 234, 236
ones, 228
pinv, 246
prod, 232
qr, 239, 245, 246
quadprog, 9
rand, 227
rcond, 242
size, 224
spy, 227
sum, 232, 234
svd, 240
trace, 232
tril, 227
triu, 227
zeros, 228
Matlab examples
popobj.m, 6
popsol.m, 6
popt.m, 8
Matrix, 223
banded, 226
diagonal, 20, 226
indefinite, 228
lower triangular, 226
unit, 226
negative definite, 228
negative semi-definite, 228
orthogonal, 228
permutation, 228
positive definite, 228
positive semi-definite, 228
rank-1, 231
skew symmetric, 228
sparse, 226
square, 228
symmetric, 16, 228
symmetric storage, 42
tridiagonal, 226
upper triangular, 226
unit, 226
Matrix factorization, 236
Cholesky, 237
LU, 237
QR, 238
SVD, 239
Matrix multiplication, 45, 247
blocked, 248
column oriented, 248
definition, 224
not commutative, 225
row oriented, 248
Matrix norm, 235
consistent, 236
Frobenius, 236
orthogonal invariance, 236
subordinate, 236
triangle inequality, 235
Mean, 254
Mean-variance models, 179
Median, 253
Merit functions, 165
Method of Lagrange multipliers, 74
Minimax problems, 30
Minimizer
global, 12
strict, 12
local, 13
strict, 13
Minimum, 13
Modelling languages
AMPL, 3
GAMS, 3
Moments, 254
Multi-objective optimization, 181
Multi-objective problems, 30
Multivariate normal distribution, 259

NAG, 249
Necessary and sufficient condition, 26
Necessary condition, 26
equality constraints
first order, 73
second order, 76
inequality constraints
first order, 85
unconstrained
first order, 63
second order, 64
Negative definite, 20, 228
Negative semi-definite, 20, 228
Neighbourhood, 13
Newton's method, 98, 120

domain of attraction, 134
fractals, 134
function minimization, 120
nonlinear equations, 134
with 1 variable, 98
Non a number `Nan`, 223
Nonlinear equations, 130
Nonsmooth, 203
Nonsmooth function, 39
Nonsmooth optimization
one-dimensional, 102
Norm
matrix, *see* Matrix norm
vector, *see* Vector norm
Normal distribution, 255
bivariate, 259
multivariate, 259
Normal equations, 132, 243
NP-complete, 44
Null space, 230
Numerical sensitivity, 242

Objective function, 29
Optimization problem
multi-variable, 29
one dimensional, 29
standard form, 4
structure, 29
Order notation
as limit $\to 0^+$, 19
as limit $\to \infty$, 42
Order of convergence, 46, 114
Orthogonal
complement, 231
matrix, 228
projection, 239, 245
vectors, 231
Orthonormal, 231
Outer product, 225
Over-determined, 130
Overflow, 38, 44

Parametric programming, 181
Partial pivoting, 237
Partially separable, 30, 36
Penalty functions, 165
ℓ_1, 166
ℓ_∞, 167
augmented Lagrangian, 167
exact, 170
sum of squares, 166, 168
Percentile, 253
Permutation matrix, 228
Polak – Ribiere method, 143
Polyhedral convex
function, 56
set, 53
Polynomial interpolation, 100
Portfolio optimization, 175
maximizing returns, 179
mean variance models, 179
minimizing risk, 181
semi-variance, 184
skewness, 185
Positive definite, 20, 228
Cholesky factorization, 238
eigenvalues, 233
principal minors, 20, 229
trace and determinant, 20
Positive homogeneous, 103
Positive semi-definite, 20, 228
Positive semidefinite, 259
Positively homogeneous, 204
Post office parcel problem, 5
Preconditioned conjugate gradient method, 144
Principal minor, 20, 229
Probabilities, 253
Probability, 253
Probability density function, 253
Problem size, 41
Pseudo inverse, 243

QR Factorization, 238
Quadratic convergence, 46
Quadratic function, 17, 21
Quadratic interpolation, 101
Quadratic models, 107
Quadratic termination, 114, 128
Quasi-Monte Carlo methods, 219
Quasi-Newton methods, 125
BFGS update, 127
Broyden family, 127
DFP update, 127
symmetric rank-1, 127
Quasi-Newton relation, 126

Random variable, 253
continuous, 253
discrete, 253
Range space, 230
Rank, 230
Rank-1 matrix, 231
Rate of convergence
linear, 46
quadratic, 46
Rate of convergence, 45
linear, 46
quadratic, 46
superlinear, 46
Ray in $\mathbb{R}^n$, 226

Regular point
  equality constraints, 73
  inequality constraints, 84
Regularization, 172
Residual, 243
Risk, 175
  efficient frontier, 177
  semi-variance, 184
  skewness, 185
  variance, 179
Rosenbrock's function, 16
Rounding error, 38

Saddle point, 64
Scenarios, 182
Secant method, 99
Semi-variance, 184
Sensitivity analysis
  equality constraints, 80
  inequality constraints, 92
Separable, 30
Sequential linear programming, 170
Sequential quadratic programming, 172
Sherman-Morrison formula, 231
Sherman-Morrison-Woodbury formula, 231
Simple bounds, 31
Simulated annealing, 13, 219
Singular value decomposition, 239
Skew symmetric matrix, 228
Skewness, 185, 254
Slope, 17
Span, 230
Sparse matrix, 34, 36, 43, 226
Spectral radius, 232
Standard deviation, 254
Standard form, 4
Stationary point, 64
Steepest ascent, 80
Steepest descent, 80, 115
Stochastic problems, 30
Strict complementarity, 89
Subdifferential, 205
Subgradient, 204
Subspace, 230
Sufficient condition, 26
  equality constraints
    second order, 76
  inequality constraints
    second order, 89
  unconstrained
    second order, 64
Sums of squares, 130
Superlinear convergence, 46
Supremum, 15

Symmetric matrix, 16, 228
Symmetric rank-1 update, 127

Taylor series, 19
Trace, 20, 232
Transpose, 224
Travelling Salesman Problem, 43, 45
Triangle inequality, 21
Trust Region, 111

Unconstrained problem, 31
Under-determined, 130
Underflow, 38, 45
Unit ball, 23, 234
Upper triangular matrix, 226
Upper-semicontinuous, 206

Variance, 254
Vector, 223
  column, 4
  row, 4
Vector norm, 22, 233
  ∞-norm, 22, 234
  1-norm, 22, 234
  2-norm, 22, 234
  A-norm, 23, 234
  p-norm, 233

Well-determined, 130

Zero testing, 38