Practical Methods of Optimization

Volume 2
Constrained Optimization

R. Fletcher
Department of Mathematics
University of Dundee, Scotland, U K

A Wiley–Interscience Publication

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto

British Library Cataloguing in Publication Data.

Fletcher, R.
Practical methods of optimization.
Vol. 2: Constrained optimization
1. Mathematical optimization
I. Title
515 QA402.5 80-42063

ISBN 0 471 27828 9

Typeset by Preface Ltd, Salisbury, Wilts, and printed in the United States of America.

Contents

Chapter 7

Introduction

7.1 Preview

The motivation for studying constrained minimization has been discussed at some length in Chapter 1 of Volume 1 of this book. The mathematical background given there, and indeed many of the concepts which arise in unconstrained optimization, are important in the study of constrained optimization. In this volume, the selection of material from the extensive literature which exists has again been done with the main theme of practicality in mind. Thus topics such as reliability and effectiveness are uppermost; to some extent these are measured by convergence and rate of convergence results. These aspects are therefore studied in some detail, and together with the subject of optimality conditions they provide good material for an academic course. However the use of *experimentation* to validate the properties of an algorithm is still of paramount importance. In fact the study of constrained optimization is by no means as well advanced as for the unconstrained case. The writing of software is a much more complex task, and so comparative experimental results are much less widely available. Often there is even a lack of suitable test problems. Also many more special cases arise and the problem of assessing numerical evidence is more difficult. For all these reasons Volume 2 departs from the feature in Volume 1 of presenting detailed numerical evidence. Nonetheless important experimental results do exist in the literature and the selection of material is guided by such results. This lack of certainty also shows up in that the decision as to precisely what algorithm to recommend in any one case is often not clear. For this reason the availability of good well-documented library software is often poor. Thus I appreciate the fact that many algorithms are necessarily used which are not ideal and I have tried to make users aware of defects in these algorithms and to enable them to mitigate their worst effects.

The structure of most constrained optimization problems is essentially contained in the following:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) && \mathbf{x} \in \mathbb{R}^n \\ &\text{subject to } c_i(\mathbf{x}) = 0, && i \in E \\ &\qquad\qquad c_i(\mathbf{x}) \geqslant 0, && i \in I. \end{aligned} \tag{7.1.1}$$

As in Volume 1, $f(\mathbf{x})$ is the *objective function*, but there are additional *constraint functions* $c_i(\mathbf{x})$, $i = 1, 2, \ldots, p$. E is the index set of equations or equality con-

straints in the problem, I is the set of inequality constraints, and both these sets are finite. More general constraints can usually be put into this form: for example $c_i(\mathbf{x}) \leqslant b$ becomes $b - c_i(\mathbf{x}) \geqslant 0$. If any point $\mathbf{x}'$ satisfies all the constraints in (7.1.1) it is said to be a *feasible point* and the set of all such points is referred to as the *feasible region* R. As in Volume 1, maximization problems are easily handled by the transformation $\max f(\mathbf{x}) = -\min -f(\mathbf{x})$. Also, a local minimizer or solution (referred to by $\mathbf{x}^*$) is looked for, rather than a global minimizer, the computation of which can be difficult. It is possible to illustrate the effect of the constraints when $n = 2$ by drawing the zero contour of each constraint function. For an equality constraint, the line itself is the set of feasible points; for an inequality constraint the line marks the boundary of the feasible region and the infeasible side is conventionally shaded. This is shown in Figure 7.1.1. Case (i) has constraints $x_2 = x_1^2$, $\mathbf{x} \geqslant \mathbf{0}$, which can be written $c_1(\mathbf{x}) = x_2 - x_1^2$, $c_2(\mathbf{x}) = x_1$, $c_3(\mathbf{x}) = x_2$ and $E = \{1\}$, $I = \{2, 3\}$. Case (ii) has constraints $x_2 \geqslant x_1^2$, $x_1^2 + x_2^2 \leqslant 1$ which can be written as $c_1(\mathbf{x}) = x_2 - x_1^2$, $c_2(\mathbf{x}) = 1 - x_1^2 - x_2^2$, $I = \{1, 2\}$, and E empty. Formulation (7.1.1) covers most types of problem; however the condition that some variables x_i take only discrete values is not included. This type of condition is covered in *integer programming* which is largely beyond the scope of this book. However a useful general purpose algorithm is the branch and bound method which enables the problem to be reduced to a sequence of smooth problems and hence solved by other techniques given in this book. This is described in Section 13.1. Another type of condition which is not included in (7.1.1) is a constraint of the form $c_i(\mathbf{x}) > 0$; something more is said about this case in Section 7.2. In fact there is often some choice in how best to pose the problem in the first instance and a number of possibilities of this type are discussed in Section 7.2.

It is assumed in (7.1.1) that the functions $c_i(\mathbf{x})$ are continuous which implies that R is closed. It is also assumed that $f(\mathbf{x})$ is continuous for all $\mathbf{x} \in R$ and preferably for all $\mathbf{x} \in \mathbb{R}^n$. If in addition the feasible region is bounded ($\exists a > 0$ such that $\| \mathbf{x} \| \leqslant a \; \forall \; \mathbf{x} \in R$), then it follows that a solution $\mathbf{x}^*$ exists. If not then the problem may be *unbounded* ($f(\mathbf{x}) \to -\infty$) or may not have a minimizing point. The problem also has no solution when R is empty, that is when the constraints are inconsistent. In fact most practical methods require the stronger assumption that the objective and constraint functions are also smooth in that their first and often second continuous derivatives exist ($f, c_i \in \mathbb{C}^1$ or $\mathbb{C}^2$). The notation ∇f (= $\mathbf{g}$) and $\nabla^2 f$ (= $\mathbf{G}$) for

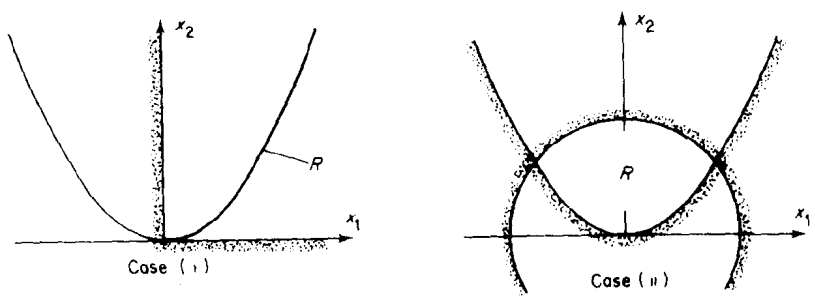

Figure 7.1.1 Examples of feasible regions

the gradient vector and Hessian matrix of f is described in Volume 1. The notation ∇c_i and $\nabla^2 c_i$ is used to denote the corresponding first and second derivatives of any constraint function c_i. The vector ∇c_i is also denoted by $\mathbf{a}_i$ and is referred to as the *normal vector* of the constraint c_i. Note that $\mathbf{a}_i$ refers to the ith vector in a set and not to the ith component of $\mathbf{a}$. These vectors are sometimes collected into columns of the *Jacobian matrix* $\mathbf{A}$ (although this rule is contradicted in the simplex method (Chapter 8) in which the normal vectors are the rows of $\mathbf{A}$). The vector $\mathbf{a}_i'$ (that is $\mathbf{a}_i(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{x}'$) is the direction of greatest increase of $c_i(\mathbf{x})$ at $\mathbf{x}'$, and if $c_i = 0$ and $i \in I$ then the direction is on the feasible side of the constraint (see Figure 7.1.2) and is at right angles to the zero contour. Most of Volume 2 assumes the existence of these derivatives which can be used, for example, to characterize optimality conditions, as described in Chapter 9, which generalize the results for unconstrained minimization given in Volume 1, Section 2.1. This is not to say necessarily that user supplied formulae for these derivatives are required in any method. Mostly, however, formulae for first derivatives are required, and in some cases formulae for second derivatives also. Methods which require no derivative information have not been studied to any great extent and the obvious advice is to estimate these derivatives by finite differences (see Volume 1), although the resulting algorithm is likely to be less robust and effective when this is done. A different situation arises when the functions f and c_i do not have continuous derivatives, which is referred to as *non-differentiable* or *non-smooth optimization*. In this case methods for smooth problems are not appropriate and special attention must be given to the surfaces of non-differentiability. These behave somewhat like the boundary of a constraint in (7.1.1) and it is therefore appropriate to discuss the problem within the structure of this volume. This is done for unconstrained non-differentiable optimization in Chapter 14: in fact it is also possible to generalize these ideas to include both smooth and non-smooth constraint functions, but this is beyond the scope of the book, although some references are given.

Another important concept is that of an *active* or *binding constraint*. Active constraints at any point $\mathbf{x}'$ are defined by the index set

$$\mathscr{A}' = \mathscr{A}(\mathbf{x}') = \{i : c_i(\mathbf{x}') = 0\} \tag{7.1.2}$$

so that any constraint is active at $\mathbf{x}'$ if $\mathbf{x}'$ is on the boundary of its feasible region. If $\mathbf{x}'$ is feasible then $\mathscr{A}' \supset E$ clearly follows. In particular the set $\mathscr{A}^*$ of active constraints at the solution of (7.1.1) is of some importance. If this set is known then

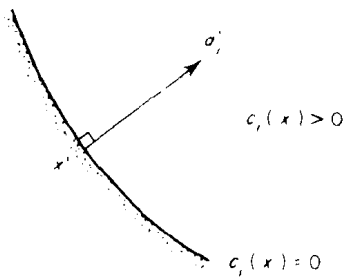

Figure 7.1.2 The normal vector

the remaining constraints can be ignored (locally) and the problem can be treated as an equality constraint problem with $E = \mathscr{A}^*$. Also constraints with $i \notin \mathscr{A}^*$ can be perturbed by small amounts without affecting the local solution whereas this is not usually true for an active constraint. An example is given by the problem: minimize $f(\mathbf{x}) = -x_1 - x_2$ subject to $x_2 \geqslant x_1^2$ and $x_1^2 + x_2^2 \leqslant 1$. Clearly from Figure 7.1.3 the solution is achieved at $\mathbf{x}^* = (1/\sqrt{2}, 1/\sqrt{2})^T$ when the contour of $f(\mathbf{x})$ is a tangent to the unit circle. Thus the active set in the notation of Figure 7.1.1(ii) is $\mathscr{A}^* = \{2\}$, and the circle constraint $c_2(\mathbf{x})$ is active. Likewise the parabola constraint $c_1(\mathbf{x})$ is inactive and can be perturbed or removed from the problem without changing $\mathbf{x}^*$. A further refinement of this definition to include strongly active and weakly active constraints is given in Figure 9.1.2.

Methods for the solution of (7.1.1) are usually iterative so that a sequence $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots$, say, is generated from a given point $\mathbf{x}^{(1)}$, hopefully converging to $\mathbf{x}^*$. If $\mathbf{x}^*$ is a member of the sequence then the method is said to *terminate*. Some early methods for constrained optimization were developed in an ad hoc way and are not strongly supported theoretically. Because of this these methods are often unreliable and expensive for problems of any size and they are not described here. However a review of what has been attempted is given by Swann (1974). The subject of constrained optimization splits into two main parts, *linear constraint programming* and *nonlinear programming* which have quite different features. In linear constraint programming each constraint is a *linear function* $c_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i$. The boundary of the feasible region for any one such constraint is a hyperplane, and the normal vector ∇c_i is constant and is again the vector $\mathbf{a}_i$. Linear constraint problems can be handled by a combination of an elimination method and an active set method (see Section 7.2) and the iterates $\mathbf{x}^{(k)}$ are always feasible points. The simplest cases are when the objective function is either linear or quadratic (*linear programming* or *quadratic programming* – Chapters 8 and 10 respectively) in both of which cases algorithms which terminate can be determined. The application to a general objective function is given in Chapter 11, and in this case many of the possibilities for unconstrained optimization in Volume 1 carry over directly. For example there are analogues of Newton's method, quasi-Newton

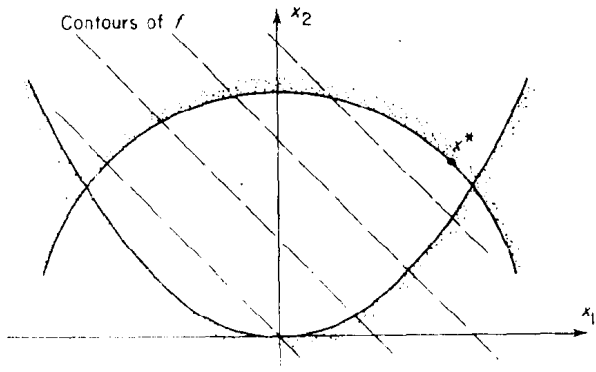

Figure 7.1.3 Active and inactive constraints

methods, the Gauss–Newton method, restricted step methods, and no-derivative methods which use finite difference approximations. Similar considerations hold in regard to using line searches and in regard to deciding what type of convergence test to use to terminate the iteration. Special cases of a linear constraint are the bounds $x_i \geqslant l_i$ or $x_i \leqslant u_i$ in which $\mathbf{a}_i$ is $\pm \mathbf{e}_i$, the ith coordinate vector, and it is particularly simple to handle such constraints. It is important that algorithms should take this into account.

The most difficult type of constrained minimization problem is nonlinear programming (Chapter 12) in which there exist some non-linear constraint functions in the problem. In this case a completely satisfactory general purpose method has yet to be agreed upon and the subject is one of intense research activity. Of course if the non-linear constraints can be rearranged so as to be eliminated directly then this should be done, but is not usually possible. Indirect elimination by solving a system of equations numerically is possible (Sections 7.2 and 12.4) but is not usually efficient and other difficulties exist; this idea is closely related to another approach known as a *feasible direction method*. A different approach is to attempt to transform the problem to one of unconstrained minimization by using *penalty functions* (Sections 12.1, 12.2, and 14.3). Efficiency depends on exactly how this is done, but it seems inevitable that some sort of penalty function must be used to get good global convergence properties. In many algorithms the iteration is determined by modelling the original problem in a suitable way. In particular a *linearization* of the constraint functions is often used. This is a first order Taylor series approximation about the current iterate $\mathbf{x}^{(k)}$:

$$c_i(\mathbf{x}^{(k)} + \boldsymbol{\delta}) \approx l_i^{(k)}(\boldsymbol{\delta}) = c_i^{(k)} + \mathbf{a}_i^{(k)\mathrm{T}}\boldsymbol{\delta}, \qquad i = 1, 2, \ldots, p. \tag{7.1.3}$$

The linearized function $l_i^{(k)}$ is defined in terms of the correction $\boldsymbol{\delta}$ to $\mathbf{x}^{(k)}$, the superscript k indicating that it is made on iteration k. This approximation enables linear constraint subproblems to be solved on each iteration. As in Volume 1, it is also possible to make a quadratic model of the objective function. However to take constraint curvature correctly into account it is appropriate to modify the quadratic term in a suitable way. Methods of this type are very important, although to obtain good global properties they must be incorporated with some type of penalty function (Sections 12.3 and 14.3). A special case of nonlinear programming is *geometric programming* in which the functions f and c_i have a polynomial type structure. It is possible to reduce this problem to a linear constraint problem which is more readily solved (Section 13.2). Often linear constraints and bounds arise in nonlinear programming problems: it is usually possible to take advantage of this fact to make the algorithm more efficient.

These algorithms, in particular those for nonlinear programming, depend on a study of optimality conditions for problem (7.1.1), and this theory is set out in Chapter 9. In Volume 1, I tried to write parts of the book in simple terms, avoiding the use of too much theory. To some extent I have done this here, for example in the presentation of linear and quadratic programming. However constrained minimization problems are much more complex than unconstrained problems and it is important for the user to have some grasp of this theory. This is especially true

in regard to *Lagrange multipliers* and *first order conditions* and I have tried to give a simple semi-rigorous introduction in Section 9.1 showing how these multipliers arise and can be interpreted. A more rigorous presentation then follows. The same is true in regard to second order conditions in Section 9.3. Some simple notions of convexity and duality for smooth problems appear in Sections 9.4 and 9.5. The subject of *non-differentiable optimization* is arguably the most difficult that I have tried to cover in this book. Some presentations of this subject are extremely theoretical although I have tried to avoid this as much as possible. However at the expense of introducing a little more theory concerning optimality conditions for non-differentiable convex functions (Section 14.2), a reasonably elegant and not too difficult treatment can be given. A discussion of algorithms (Sections 14.4 and 14.5) is also given.

7.2 Elimination and Other Transformations

There is considerable scope for making transformations to a constrained minimization problem which reduce it to a form which is more readily solved. This can be advantageous and a number of such possibilities are discussed. However it is important to be aware from the outset that this procedure is not entirely without risk and that solutions of the original and transformed problems may not correspond on a one-to-one basis or that methods may not perform adequately on the transformed problem. A number of examples of this are given in this section and elsewhere in the book, and the user should be on his guard. The most simple possibility for equality constraints is to use the equations to eliminate some of the variables in the problem (*elimination*). If there are just m equations $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ which can be rearranged directly to give

$$\mathbf{x}_1 = \boldsymbol{\phi}(\mathbf{x}_2) \tag{7.2.1}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are partitions of $\mathbf{x}$ in $\mathbb{R}^m$ and $\mathbb{R}^{n-m}$, then the original objective function $f(\mathbf{x}_1, \mathbf{x}_2)$ is replaced by

$$\psi(\mathbf{x}_2) = f(\boldsymbol{\phi}(\mathbf{x}_2), \mathbf{x}_2) \tag{7.2.2}$$

and $\psi(\mathbf{x}_2)$ is minimized over $\mathbf{x}_2$ without any constraints. A simple example is given in Question 7.3. Derivatives of ψ are readily obtained from those of f and $\mathbf{c}$ (see Question 7.5). Some care has to be taken to avoid an ill-conditioned rearrangement when forming (7.2.1), for instance with linear constraints it is advisable to use some sort of pivoting on the variables. In some cases the method may fail completely, as shown in Question 7.4. In fact it is possible to discuss elimination in more general terms, implicitly by first making a linear transformation of variables; this is described in Section 10.1. In cases where no direct rearrangement like (7.2.1) is available, it is possible to regard $\mathbf{c}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{0}$ as a system of non-linear equations which can be solved by the Newton–Raphson method (Volume 1, Section 6.2). In doing this $\mathbf{x}_2$ remains fixed and a vector $\mathbf{x}_1$ is determined which solves the equations. Thus $\mathbf{x}_1$ depends on $\mathbf{x}_2$ and so the process implicitly defines a function $\mathbf{x}_1 = \boldsymbol{\phi}(\mathbf{x}_2)$. This method is outlined in a more general form in Section 12.4;

however the process is not always the most efficient and there can be difficulties in getting the Newton–Raphson method to converge. An alternative transformation for the equality constraint problem is the *method of Lagrange multipliers* (Section 9.1) in which the system of non-linear equations (9.1.5) is solved which arises from the first order necessary conditions. Except in special cases this system must be solved numerically, which renders the method of little practical use. The method can also fail, not only when the solution of (9.1.5) corresponds to a constrained maximum point or saddle point, but also when the regularity condition (9.2.4) does not hold (see Question 9.14).

Elimination methods are not directly applicable to inequality constraint problems unless the set of active constraints $\mathscr{A}^*$ is known. However it is possible to use a trial and error sort of method in which a guess $\mathscr{A}$ is made at the set of active constraints, and constraints in $\mathscr{A}$ are then treated as equalities, neglecting the remaining inequality constraints. The resulting equality constraint problem is then solved by elimination or by the method of Lagrange multipliers, giving a solution $\hat{\mathbf{x}}$. It is necessary to check that $\hat{\mathbf{x}}$ is feasible with respect to the constraints which have been ignored. If not, one of these is added to the active set and the above process is repeated. If $\hat{\mathbf{x}}$ is feasible then it is also necessary to check that the first order conditions are satisfied. To do this requires the calculation of the corresponding Lagrange multiplier vector $\hat{\boldsymbol{\lambda}}$. Since $\lambda_i = \partial f/\partial c_i$ to first order measures the effect of perturbations in the c_i on f, it is necessary for an inequality constraint $c_i(\mathbf{x}) \geqslant 0$ that $\lambda_i \geqslant 0$ at the solution, for otherwise a feasible perturbation would reduce f. Thus if there are any $\hat{\lambda}_i < 0$, one such constraint must be removed from the active set and the process repeated again. On the other hand, if $\hat{\boldsymbol{\lambda}} \geqslant \mathbf{0}$ then the required solution is located. Methods of this type can be used in an informal way on small problems. However they are most useful in solving all types of linear constraint problem when systematic procedures can be devised. Such methods include the *simplex method* for linear programming and the *active set method* for all types of linear constraint programming. So-called *exchange algorithms* for best linear L_1 and L_∞ data fitting are also examples of this type of procedure. Systematic procedures using active set methods for non-linear constraints based on solving the equality constraint problems by implicit elimination can also be devised (Section 12.4) but there are some difficulties which are not readily overcome. It is difficult to handle constraints of the form $c_i(\mathbf{x}) > 0$ in an active set method because the feasible region is not closed and the constraint cannot be active at a solution. However it can be useful to include them in the problem via the transformation $c_i(\mathbf{x}) \geqslant \epsilon > 0$, possibly solving a sequence of problems in which $\epsilon \downarrow 0$ if the constraints happen to be active. The reason for doing this might be to prevent or dissuade $f(\mathbf{x})$ being evaluated at an infeasible point at which it is not defined (for example the problem: $\min x \log_e x$ subject to $x > 0$). It may not be satisfactory just to ignore the constraints because the problem may then become unbounded or have a global solution with $c_i(\mathbf{x}) \leqslant 0$, which is of no interest.

Some other transformations are worthy of note which relate equality and inequality constraint problems. For example a constraint $c_i(\mathbf{x}) = 0$ can be equivalently replaced by two opposite inequality constraints $c_i(\mathbf{x}) \geqslant 0$ and $-c_i(\mathbf{x}) \geqslant 0$.

This enables (7.1.1) to be reduced to an inequality constraint problem. However there are some practical disadvantages due to degeneracy and other reasons and the idea is best avoided, although it can occasionally be useful. The alternative possibility is to write $c_i(\mathbf{x}) \geqslant 0$ as the equality constraint $\min(c_i(\mathbf{x}), 0) = 0$. Unfortunately this function is not a $\mathbb{C}^1$ function and so is usually excluded on this count. Another possibility is to replace a constraint $c_i(\mathbf{x}) \geqslant 0$ by adding an extra variable, z say, giving an equality constraint $c_i(\mathbf{x}) = z$ and a bound $z \geqslant 0$. The variable z is referred to as a *slack variable* since it measures the slack in the inequality constraint. This transformation is most useful in the simplex method for linear programming which requires all general inequality constraints to be handled in this way, but is not necessary in active set methods which treat inequalities of any type directly. Furthermore, following an idea introduced later in this section, it is possible to do away with the need for the bound $z \geqslant 0$. This is done by adding a *quadratic slack variable* y and replacing $c_i(\mathbf{x}) \geqslant 0$ by the (non-linear) equality constraint $c_i(\mathbf{x}) = y^2$. This removes the need to treat inequality constraints directly. However this transformation does cause some distortion as explained below and in this case it may be somewhat dangerous, in particular because of the following feature. Let for example $c(\mathbf{x}) \geqslant 0$ be the only constraint and let $\mathbf{x}'$ be such that $c' = 0$ and $\mathbf{g}' = \mathbf{a}'\lambda'$ where $\lambda' < 0$. Then $\mathbf{x}'$ and λ' do not satisfy first order conditions for a solution. Yet the vector $\mathbf{x}'$ augmented by $y' = 0$ does satisfy first order conditions in the transformed equality constraint problem with the same λ'. Thus the transformation does not seem to be able to distinguish whether or not constraints are active on the basis of first order information. I have also heard bad reports of quadratic slacks in practice which might well be accountable for in this way.

Many other useful transformations arise in constrained optimization and are used in subsequent chapters. Perhaps the most well-known idea is the use of *penalty functions* for nonlinear programming. The idea is to transform the problem to one of unconstrained optimization by adding to the objective function a penalty term which weights constraint violations. In *sequential penalty functions* $\mathbf{x}^*$ is found as the limit of the minimizing points of a sequence of penalty functions, as some controlling parameter is changed. More recently the value has been realized of an *exact penalty function* which has $\mathbf{x}^*$ as its local minimizer. These transformations are described in some detail in Sections 12.1, 12.2, 12.5, and 14.3. Other transformations of some importance are those arising in duality (Section 9.5), integer programming (Section 13.1), and geometric programming (Section 13.2), amongst others.

It is also possible to make transformations of variables in an attempt to simplify the problem. For example the bound $x_i \geqslant 0$ can be removed by defining a new variable y_i which replaces x_i, such that $x_i = y_i^2$. Then for any y_i in $(-\infty, \infty)$ it follows that $x_i \geqslant 0$ so the bound does not need to be explicitly enforced. Another similar transformation for $\ell_i \leqslant x_i \leqslant u_i$ is to let y_i satisfy $x_i = \ell_i + (u_i - \ell_i)\sin^2 y_i$. For strict constraints $x_i > 0$ it is possible to use $x_i = e^{y_i}$. The advantage of these transformations is that they do extend the range of problems which can be handled by an unconstrained minimization routine. This is not to say that minimization with simple bounds $\ell_i \leqslant x_i \leqslant u_i$ is at all difficult; in fact the opposite is true and

it is probably more efficient to treat the problem directly. It is simply that subroutines which minimize functions subject only to bounds are much less readily available to the user at present. These ideas can also be used to transform inequality constraints to equalities (see above in regard to quadratic slacks), although this possibility should be viewed with some suspicion for the reason given above. These transformations do cause some distortion which often may not be favourable. For example the problem $\min x^2$ subject to $x \geqslant 0$, after transforming $x = y^2$, becomes $\min y^4$. This has a singular Hessian at the solution which causes any standard minimization method based on a quadratic model to converge slowly. Another example is the convex programming problem $\min(x-1)^2$ subject to $x \geqslant 0$. Although the transformation is well behaved at the solution $x^* = 1$, it induces a stationary point with a non-positive-definite Hessian matrix at $x = 0$ and both these features could possibly cause difficulties (see also Question 7.6). Thus although such transformations can be useful, the user should be aware that they are not entirely risk free.

Another transformation which enables $|x_i|$ functions to be handled is to replace the variable x_i by two non-negative variables x_i^+ and x_i^- representing the positive and negative parts of x_i (that is $\max(x_i, 0)$ and $\max(-x_i, 0)$). The conditions $x_i^+ \geqslant 0$ and $x_i^- \geqslant 0$ are explicitly included in the problem; also whenever x_i appears in the problem it is replaced by $x_i^+ - x_i^-$ and similarly $|x_i|$ is replaced by $x_i^+ + x_i^-$ (see Question 8.12). This latter replacement is only valid if one of x_i^+ or x_i^- is zero, which can sometimes be guaranteed, for example in otherwise linear problems when both x_i^+ and x_i^- together cannot be basic (Chapter 8). This transformation can also be used to handled unbounded variables in a linear programming problem. An alternative technique for handling $|x_i|$ terms is described in Section 8.4. These ideas can be extended to functions $|c_i(x)|$ by adding extra variables y_i and the equality constraint $c_i(\mathbf{x}) = y_i$ (see Question 8.11), thus enabling L_1 approximation problems to be handled by smooth techniques. Similar ideas for minimizing max functions or L_∞ functions can be tackled by introducing an extra variable v as described in Section 14.1. However all these techniques are really attempting to solve non-differentiable optimization problems as smooth problems. In the current state of the art this can be useful, but when software becomes readily available for some of the better more direct methods described in Sections 14.4 and 14.5, these should be preferred.

Finally the very important transformation of *scaling* either the constraints or the variables in the problem is discussed. Scaling of a constraint set is achieved by multiplying each constraint function by a constant chosen so that the value of each constraint function, evaluated for typical values of $\mathbf{x}$, is of the same order of magnitude. This can be important in that this scales the Lagrange multipliers (inversely) and so can make more reliable the test on the magnitude of a multiplier which is used in some algorithms. A well-scaled matrix is also important in some linear algebra routines when pivoting tests are made. Moreover when using penalty functions which involve quantities like $\mathbf{c}^T\mathbf{c}$ or $\|\mathbf{c}\|_1$, it is important that constraints are scaled. In a similar way scaling of the variables can sometimes be important. This again arises when pivoting tests on the variables are made, or when implicitly using

some norm of the variables, for example in restricted step methods or methods with a bias towards steepest descent (see Volume 1). In practice variables are usually scaled by multiplying each one by a suitable constant. However a non-linear scaling which can be useful for variables $x_i > 0$ is to use the transformation $x_i = e^{y_i}$. Then variables of magnitudes $10^{-6}, 10^{-3}, 10^{0}, 10^{3}, \ldots$, say, which typically can occur in kinetics problems, are transformed into logarithmic variables with magnitudes which are well scaled.

Questions for Chapter 7

1. Calculate the Jacobian matrix $\nabla \ell^{\mathrm{T}}$ of the linear system $\ell(\mathbf{x}) = \mathbf{A}^{\mathrm{T}}\mathbf{x} - \mathbf{b}$. If ℓ is obtained by linearizing a non-linear system as in (7.1.3) show that both systems have the same Jacobian matrix.
2. Obtain the gradient vector and the Hessian matrix of the functions $f(\mathbf{x}) + h(\mathbf{c}(\mathbf{x}))$ and $f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{c}(\mathbf{x})$. In the latter case treat the cases both where $\boldsymbol{\lambda}$ is a constant vector and where it is a function $\boldsymbol{\lambda}(\mathbf{x})$.
3. Find the solution of the problem minimize $-x - y$ subject to $x^2 + y^2 = 1$ by graphical means and also by eliminating x, and show that the same solution is obtained. Discuss what happens, however, if the square root which is required is chosen to have a negative sign.
4. Solve the problem minimize $x^2 + y^2$ subject to $(x-1)^3 = y^2$ both graphically and also by eliminating y. In the latter case show that the resulting function of x has no minimizer and explain this apparent contradiction. What happens if the problem is solved by eliminating x?
5. Consider finding derivatives of the functions $\boldsymbol{\phi}(\mathbf{x}_2)$ and $\psi(\mathbf{x}_2)$ defined in (7.2.1) and (7.2.2). Define partitions

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{pmatrix}, \qquad \mathbf{g} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{pmatrix}, \qquad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}$$

   and show by using the chain rule that $\nabla_2 \boldsymbol{\phi}^{\mathrm{T}} = -\mathbf{A}_2\mathbf{A}_1^{-1}$ and hence that $\nabla_2 \psi = \mathbf{g}_2 - \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{g}_1$. Second derivatives of ψ are most conveniently obtained as in (12.4.6) and (12.4.7) by setting $\mathbf{V}^{\mathrm{T}} = [\mathbf{0} : \mathbf{I}]$ and hence $\mathbf{Z}^{\mathrm{T}} = [-\mathbf{A}_2\mathbf{A}_1^{-1} : \mathbf{I}]$ and $\mathbf{S}^{\mathrm{T}} = [\mathbf{A}_1^{-1} : \mathbf{0}]$.
6. Consider the problem minimize $f(x_1, x_2)$ subject to $x_1 \geqslant 0, x_2 \geqslant 0$ when the transformation $x = y^2$ is used. Show that $\mathbf{x}' = \mathbf{0}$ is a stationary point of the transformed function, but is not minimal if any $g_i' \leqslant 0$. If $\mathbf{g}' = \mathbf{0}$ then second order information in the original problem would usually enable the question of whether $\mathbf{x}'$ is a minimizer to be determined. Show that in the transformed problem this cannot be done on the basis of second order information.

Chapter 8
Linear programming

8.1 Structure

The most simple type of constrained optimization problem is obtained when the functions $f(\mathbf{x})$ and $c_i(\mathbf{x})$ in (7.1.1) are all linear functions of $\mathbf{x}$. The resulting problem is known as a *linear programming* (LP) problem. Such problems have been studied since the earliest days of electronic computers, and the subject is often expressed in a quasi-economic terminology which to some extent obscures the basic numerical processes which are involved. This presentation aims to make these processes clear, whilst retaining some of the traditional nomenclature which is widely used. One main feature of the traditional approach is that linear programming is expressed in the *standard form*

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}) \triangleq \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{x} \geqslant \mathbf{0}, \end{aligned} \tag{8.1.1}$$

where $\mathbf{A}$ is an $m \times n$ matrix, and $m \leqslant n$ (usually $<$). The symbol $\triangleq$ means 'defined by'. Thus the allowable constraints on the variables are either linear equations or non-negativity bounds. The coefficients $\mathbf{c}$ in the linear objective function are often referred to as *costs*. An example with four variables ($n = 4$) and two equations ($m = 2$) is

$$\begin{aligned} &\text{minimize } x_1 + 2x_2 + 3x_3 + 4x_4 \\ &\text{subject to } x_1 + x_2 + x_3 + x_4 = 1 \\ &\qquad\qquad x_1 + x_3 - 3x_4 = \tfrac{1}{2}, \\ &x_1 \geqslant 0,\ x_2 \geqslant 0,\ x_3 \geqslant 0,\ x_4 \geqslant 0. \end{aligned} \tag{8.1.2}$$

More general LP problems can be reduced to standard form without undue difficulty, albeit with some possible loss of efficiency. For instance a general linear inequality $\mathbf{a}^{\mathrm{T}}\mathbf{x} \leqslant b$ can be transformed using a slack variable z (see Section 7.2) to the equation $\mathbf{a}^{\mathrm{T}}\mathbf{x} + z = b$ and the bound $z \geqslant 0$. Alternatively the dual transformation can sometimes be used advantageously to obtain a standard form and this is described in more detail in Section 9.5. More general bounds $x_i \geqslant \ell_i$ can be dealt with by a shift of origin, and if no bound exists at all on x_i in the original problem, then the standard form can be reached by introducing non-negative variables x_i^+

and x_i^-, as described in Section 7.2. In fact very little is lost in complexity if the bounds in (8.1.1) are expressed as

$$\ell \leqslant \mathbf{x} \leqslant \mathbf{u}, \tag{8.1.3}$$

as Question 8.8 illustrates. The merit of using other possible standard forms is discussed in more detail in Section 8.3. However for the most part this text will concentrate on the solution of problems which are already in the standard form (8.1.1).

It is important to realize that a problem in standard form may have no solution, either because there is no feasible point (the problem is *infeasible*), or because $f(\mathbf{x}) \to -\infty$ for $\mathbf{x}$ in the feasible region (the problem is *unbounded*). However it is shown that there is no difficulty in detecting these situations, and so the text concentrates on the usual case in which a solution exists (possibly not unique). It is also convenient to assume that the equations are independent, so that they have no trivial linear combination. In theory this situation can always be achieved, either by removing dependent equations or by adding artificial variables (see Section 8.4 and Question 8.21), although in practice there may be numerical difficulties if this dependence is not recognized.

If (8.1.1) is considered in more detail, it can be seen that if $m = n$, then the equations $\mathbf{Ax} = \mathbf{b}$ determine a unique solution, and the objective function $\mathbf{c}^T\mathbf{x}$ and the bounds $\mathbf{x} \geqslant \mathbf{0}$ play no part. In most cases however $m < n$, so that the system $\mathbf{Ax} = \mathbf{b}$ is underdetermined and $n - m$ degrees of freedom remain. In particular the system can determine only m variables, given values for the remaining $n - m$ variables. For example the equations $\mathbf{Ax} = \mathbf{b}$ in (8.1.2) can be rearranged as

$$\begin{aligned} x_1 &= \tfrac{1}{2} - x_3 + 3x_4 \\ x_2 &= \tfrac{1}{2} \qquad - 4x_4 \end{aligned} \tag{8.1.4}$$

which determines x_1 and x_2 given values for x_3 and x_4, or alternatively as

$$\begin{aligned} x_1 &= \tfrac{7}{8} - \tfrac{3}{4}x_2 - x_3 \\ x_4 &= \tfrac{1}{8} - \tfrac{1}{4}x_2 \end{aligned} \tag{8.1.5}$$

which determines x_1 and x_4 from x_2 and x_3, and in other ways as well. It is important to consider what values these remaining $n - m$ variables can take in the standard form problem. The objective function $\mathbf{c}^T\mathbf{x}$ is linear and so contains no curvature which can give rise to a minimizing point. Hence such a point must be created by the conditions $x_i \geqslant 0$ becoming active on the boundary of the feasible region. For example if (8.1.5) is used to eliminate the variables x_1 and x_4 from the problem (8.1.2), then the objective function can be expressed as

$$f = x_1 + 2x_2 + 3x_3 + 4x_4 = \tfrac{11}{8} + \tfrac{1}{4}x_2 + 2x_3. \tag{8.1.6}$$

Clearly this function has no minimum value unless the conditions $x_2 \geqslant 0, x_3 \geqslant 0$ are imposed, in which case the minimum occurs when $x_2 = x_3 = 0$. An illustration is given in Figure 8.1.1 for the more simple conditions $x_1 + 2x_2 = 1$ and $x_1 \geqslant 0$, $x_2 \geqslant 0$. The feasible region is the line joining the points $\mathbf{a} = (0, \tfrac{1}{2})^T$ and $\mathbf{b} = (1, 0)^T$. ∎

When the objective function $f(\mathbf{x})$ is linear the solution must occur at either $\mathbf{a}$ or $\mathbf{b}$

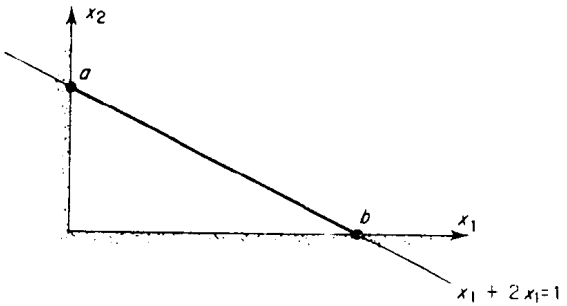

Figure 8.1.1 Constraints for a simple LP problem

with either $x_1 = 0$ or $x_2 = 0$ (try different linear functions, for example $f = x_1 + x_2$ or $f = x_1 + 3x_2$). If however $f(\mathbf{x}) = x_1 + 2x_2$ then any point on the line segment is a solution and this includes both a and b. This corresponds to the existence of a non-unique solution.

To summarize therefore, a solution of an LP problem in standard form always exists at one particular *extreme point* or *vertex* of the feasible region, with at least $n - m$ variables having *zero value*, and the remaining m variables being uniquely determined by the equations $\mathbf{Ax} = \mathbf{b}$ and taking non-negative values. This result is fundamental to the development of LP methods, and can be established rigorously using the notions of convexity (Section 9.4). The proof is sketched out in some detail in Questions 9.20 to 9.22.

The main difficulty in linear programming is to find which $n - m$ variables take zero value at the solution. The earliest method for solving this problem is the *simplex method*, which tries different sets of possibilities in a systematic way. This method is described in Section 8.2 and is still predominant today, albeit often in more sophisticated forms. Different variations of the method exist, depending upon exactly which intermediate quantities are computed. The earliest *tableau form* became superseded by the more efficient *revised simplex method*, both of which are described in Section 8.2. More recently methods based on using matrix factorizations have been suggested in order to control round-off errors more effectively. For large sparse LP problems, *product form methods* have enabled problems of up to 10^5 variables to be solved in practice. Both these developments are described in Section 8.5. An apparently different approach to LP is the active set method described in Section 8.3 which, however, turns out to be equivalent to the simplex method with slack variables, although different intermediate matrices are stored. The problem of calculating initial feasible points for LP and other linear constraint problems is described in Section 8.4. All these methods have one possible situation in which they can fail to solve a problem which has a well-defined solution. This is referred to as degeneracy and is described in Section 8.6.

8.2 The Simplex Method

The simplex method for solving an LP problem in standard form generates a sequence of feasible points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ which terminates at a solution. Since