Numerical Analysis Group Internal Report 2006-1

An introduction to algorithms for continuous optimization© [1,2]

Nicholas I. M. Gould[3,4,5], Raphael Hauser[6] Sven Leyffer[7,8]

ABSTRACT

We provide a concise introduction to modern methods for solving continuous nonlinear optimization problems. We consider both linesearch and trust-region methods for unconstrained minimization, interior-point methods for problems involving inequality constraints, and SQP methods for those involving equality constraints. Theoretical as well as practical aspects are emphasised. We conclude by giving a personal view of some of the most significant papers in the area, and a brief guide to on-line resources.

---

[1] These notes have been written to accompany a course on continuous optimization given by NIMG to undergraduate students at Oxford University in Hilary term 2006. They have been influenced by earlier courses by RH at Oxford in previous years, and that by NIMG and SL at the 10th EPSRC/LMS Numerical Analysis Summer School at the University of Durham, 15-19th July 2002. The latter appeared in the book "Frontiers in Numerical Analysis (Durham 2002)", J. F. Blowey, A. W. Craig and T. Shardlow, eds., Springer Verlag (2003) 109-197.

[2] Some of this material appeared, in expanded form, in the book "Trust-region methods" [SIAM, Philadelphia, 2000] by Andy Conn, Philippe Toint and the first author. We are grateful to both Andy and Philippe for their contributions.

[3] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU. Email: n.gould@rl.ac.uk

[4] Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.html".

[5] This work was supported in part by the EPSRC grants GR/R46641 and GR/S42170.

[6] Oxford University Computing Laboratory, Numerical Analysis Group, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, UK. Email: hauser@comlab.ox.ac.uk

[7] Mathematics and Computer Science Division,
Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA.
Email: leyffer@mcs.anl.gov

[8] Current reports available from "http://www-unix.mcs.anl.gov/∼leyffer/".

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

February 14, 2006.

Contents

GLOSSARY OF SYMBOLS

m	number of constraints
n	number of variables
x	mimimization variables
y	Lagrange multipliers
$f(x)$	objective function
$g(x)$	gradient of objective function
$H(x)$	Hessian matrix of the objective function
$c_i(x)$	i-th constraint function
$a_i(x)$	gradient of the i-th constraint function
$H_i(x)$	Hessian matrix of i-th constraint function
$c(x)$	vector of constraint functions
$A(x)$	Jacobian matrix of constraint functions
$\ell(x, y)$	Lagrangian function
$g(x, y)$	gradient of the Lagrangian function
$H(x, y)$	Hessian of the Lagrangian function
I	identity matrix
e_i	i-th column of the identity matrix
$\langle u, v \rangle$	Euclidean inner product between the generic vectors u and v
$\lambda_i(M)$	i-th largest eigenvalue of the generic matrix M
$\gamma(x)$	Lipschitz constant at x
$\mathcal{C}$	set of feasible points
$\mathcal{A}(x)$	active set at x
$\mathcal{I}(x)$	inactive set at x
$\mathcal{W}$	working set
$\mathcal{N}$	subspace of weak linearized feasible directions
$\mathcal{N}_+$	cone of linearized feasible directions
p	search direction
α	steplength
s	step
$m(s)$	step-selection model
$q(s)$	quadratic model
B	Hessian approximation
Δ	trust-region radius
ρ	ratio of actual to predicted reduction
μ	penalty/barrier parameter
$\Phi(x, \mu)$	quadratic penalty/logarithmic barrier function
$\Phi(x, u, \mu)$	augmented Lagrangian function
$y(x, \mu)$	first-order Lagrange multiplier estimate
$y(x, u, \mu)$	first-order augmented Lagrange multiplier estimates
Y, C	diagonal matrices of entries of vectors y, c

INTRODUCTION

The solution of (nonlinear) optimization problems—that is the minimization or maximization of an objective function involving unknown parameters/variables in which the variables may be restricted by constraints—or nonlinear programming as it sometimes known, is one of the core components of computational mathematics. Nature (and mankind) loves to optimize, and the world is far from linear. In his book on Applied Mathematics, the eminent mathematician Gil Strang[1] opines that optimization, along with the solution of systems of linear equations, and of (ordinary and partial) differential equations, is one of the three cornerstones of modern applied mathematics.

Optimization problems can broadly be described as either continuous or discrete, but may be a mix of both. *Discrete* optimization is, as its name suggests, concerned with the case where the variables may only take on discrete (and typically integer) values. Often these problems are very hard, and only enumeration of all possible points is guaranteed to work. Fortunately, sometimes the problems are easy, and simple (greedy) heuristics are guaranteed to work. By contrast, the variables in *continuous* optimization problems are allowed to take on any values permitted by the constraints. Here we shall only be concerned with continuous optimization. More especially, we shall restrict ourselves to problems whose defining functions are differentiable, since then we will be able to predict how small changes in variable values will affect the objective and constraints. There are good methods for non-differentiable optimization, but these often rely on estimates of (often unknown) Lipschitz constants.

[1] G. Strang, “Introduction to Applied Mathematics”, Wellesley-Cambridge Publishers (1986).

This set of notes is partitioned in broadly the same way as the courses on which it was based. Optimality conditions play a vital role in optimization, both in the identification of optima, and in the design of algorithms to find them. We consider these in Part 1. Parts 2 and 3 are concerned with the two main techniques for solving unconstrained optimization problems. Although it can be argued that such problems arise relatively infrequently in practice (nonlinear fitting being a vital exception), the underlying linesearch and trust-region ideas are so important that it is best to understand them first in their simplest setting. The remaining four parts cover the problems we really wish to solve, those involving constraints. We purposely consider inequality constraints (alone) in two of the partsand equality constraints (alone) in the other two, since then the key ideas may be developed without the complication of treating both kinds of constraints at once. Of course, real methods cope with both, and suitable algorithms will be hybrids of the methods we have considered.

Courses based on these notes have been given to both undergraduate and graduate students in Oxford, to graduates in Edinburgh and to attendees on an EPSRC Numerical Analysis Summer School in Durham. These course have been of different lengths, and each time we have had to be selective with the material taught. For a 16 lecture undergraduate course, we use most of the material, although we leave some proofs as homework exercises. For a 10–12 lecture graduate course, we normally skip the material on linearly-constrained optimization (Part 4) and penalty and augmented Lagrangian methods (Part 5), although we do need to include Section 5.1 on merit functions as a precursor to Part 6. If time is tight, conjugate-gradient methods (Sections 2.5.4, 2.5.5 and 3.6) are also sacrificed. Any perspective teacher is more than willing to use our slides if they so wish.

We make no apologies for mixing theory in with algorithms, since (most) good algorithms have good theoretical underpinnings. So as not to disturb the development in the main text, the proofs of stated theorems have been relegated to Appendix C. In addition, we do not provide citations in the main text, but have devoted Appendix A to an annotated bibliography of what we consider to be essential references in nonlinear optimization. Such a list is, by its nature, selective, but we believe that the given references form a corpus of seminal work in the area, which should be read by any student interested in pursuing a career in optimization.

Before we start, we feel that one key development during the last ten years has done more to promote the use of optimization than possibly any other. This is NEOS, the Network Enabled Optimization Server, at Argonne National Laboratory and Northwestern University in Chicago, see `http://www-neos.mcs.anl.gov/neos` . Here, users are able to submit problems for remote solution, without charge, by a large (and expanding) collection of the world's best optimization solvers, many of them being only available otherwise commercially. Further details of what may be found on the World-Wide-Web are given in Appendix B.

An example—the optimization of a high-pressure gas network

Before we embark on out tour of optimization methods, we first set the scene by considering a typical "real-world" example. In the UK (and elsewhere in the world), natural gas is extracted from the (North) sea and pumped to where is needed–the large centers of population—via a network of high-pressure pipes. This so-called National Transmission System (NTS) is the high pressure part of the UK National Grid's transmission system, and consists of more than 6,600 Kilometer's of top-quality welded steel pipeline operating at pressures of up to 85 times normal atmospheric pressure. The gas is pushed through the system using 26 strategically placed compressor stations. From over 140 off-take points, the NTS supplies gas to 40 power stations, a small number of large industrial consumers and the twelve Local Distribution Zones that contain pipes operating at lower pressure which eventually supply the consumer. We illustrate the NTS in Figure 1.

Figure 1: The National Grid Gas National Transmission System.

We may view an idealized gas network as a collection of pipes connected at nodes. In order to understand the basic equations we shall develop, we first note that the main characteristics are that we shall associate a (gas) pressure p_i with each node on the network, and a flow q_j along each pipe.

At each node, we will must have a balance equation which says that what flows in must flow out again. Identical equations occur in other networks—for electrical networks, these are the famous Kirkoff laws. We illustrate one such equation in Figure 2.

In general, for the whole network, the nodes gives us constraints

$$Aq - d = 0.$$

These are linear equations, they are sparse—each row only involves the pipes entering or leaving that particular node—and structured—the matrix A has a ± 1 and 0 structure typical of network problems.

For each pipe, we have a different kind of equation. Considerable empirical experience with thousands of real-life pipes has lead engineers to believe that the square of the pressure loss between the two end of the pipe is proportional to the 2.8359-th power of the flow along in the pipe. A pipe equation is illustrated in Figure 3.

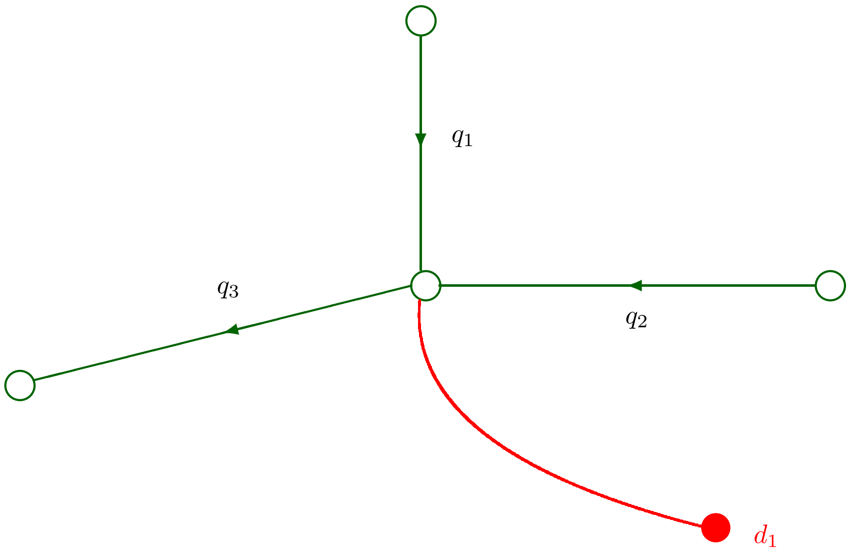

Figure 2: A typical node where there are two input flows q_1 and q_2, one output flow q_3 and an extracted demand d_1. The relevant node equation is then $q_1 + q_2 - q_3 - d_1 = 0$.

In general, for the whole network, the nodes gives us constraints

$$A^T p^2 + K q^{2.8359} = 0$$

where we have slightly abuses notation by writing $A^T p^2$ to represent the $p_{\text{out}}^2 - p_{\text{in}}^2$ term, and where K is a diagonal matrix of pipe properties. Now these are nonlinear equations, they are sparse—each row only involves the pressures at the ends of that particular pipe—and structured—the matrix A^T has a ± 1 and 0 structure typical of network problems, and is the transpose of that occurring in the node equations (you might wish to think why this should be the case).

We also need to model the compressors that drive the gas through the network. A compressor is a machine that is used to boost flow, and without these the flow would gradually grind to a halt. Compressors are expensive to run, and are usually only operated if they need to be. Without giving details, the flow through an operational compressor will be boosted by some nonlinear function of the input/output flows and pressures. If the machine is off, the flow in and out will simply be as if the machine is a node. We illustrate a typical compressor equation in Figure 4.

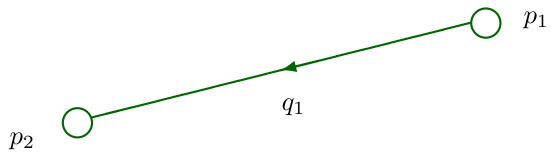

Figure 3: A typical pipe where the pressures at the start and end of the pipe are p_1 and p_2, and the flow along the pipe is q_1. The relevant pipe equation is then $p_2^2 - p_1^2 + k_1 q_1^{2.8359} = 0$, where k_1 is a constant representing properties of the pipe in question.

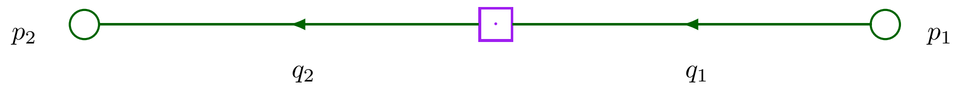

Figure 4: A typical compressor for which the pressures and flows at the entrance and exit of the machine are (p_1, q_1) and (p_2, q_2) and where the zero-one variable z_1 is nonzero if the compressor is switched on. In this case we have $q_1 - q_2 + z_1 \cdot c_1(p_1, q_1, p_2, q_2) = 0$, for some nonlinear compressor function c_1.

In general, for the whole network, the compressors gives us constraints

$$A_2^T q + z \cdot c(p, q) = 0.$$

Again these are nonlinear, often highly so, they are sparse and structured, but now they introduce the extra ingredient of zero-one (integer) variables, z.

Finally, there will be other, more obvious constraints. In particular there should be simple bounds

$$\begin{aligned} p_{\min} \leq p \leq p_{\max} \\ q_{\min} \leq q \leq q_{\max} \end{aligned}$$

on the pressures and flows for safety reasons; low pressures/flows may lead to build up of other, undesirable gases, high pressures might cause a pipe to fracture, and rapid flows may gradually ware down the surface of the pipe.

So much for the physics. The equations we have expressed so far have many solutions, and of these a big company like the UK National Grid will aim to find a solution for which some objective is best fulfilled. There are many such possibilities, of which minimizing or maximizing—yes conflicts can arise—the sum of pressures, minimizing compressor fuel costs and minimizing supply have all been proposed. In practice it is not uncommon to aim to optimize more than one objective, and sometimes formulations will try to combine these is some way.

In reality, the UK National Grid Gas system comprises roughly 200 nodes and pipes, and 26 machines. Thus the steady-state problem involves somewhere in the region of 400 unknowns. However, as the reader is undoubtedly aware, gas demand varies throughout the day and night—the sudden rush to boil a kettle half-way though a popular soap opera or the more predictable morning and evening central-heating peaks. If this variable demand is taken into account, and the pressures and demands required every 10 minutes for a whole 24-hour period, the problem grows dramatically in size to be of the order of 58,000 variables. And the challenge is to be able to solve this in real-time if, for example, there is a sudden need to redistribute gas to repair a leaking pipe.

This problem is typical of real-world, large-scale applications. It involves simple bounds on the variables and linear and nonlinear constraints. It is highly structured (this is fortunate, as otherwise we would have little chance of solving it), and the global solution is "required"—in practice, a 10% improvement potentially leads to many millions of pound in savings, so this is often deemed good enough! To cope with variable demands, the problem is actually a discretisation of a continuous one. And finally, there are integer variables, which makes the problem far harder to solve—even if we don't touch on this here.

Some other application areas

It should come as no surprise that similar problems arise in electrical-power scheduling. The variables there are potential differences and currents. As we have mentioned Kirkoff's laws constrain currents at network nodes, while currents flow along connecting cables due to potential differences between nodes. There are losses due to heat, and obvious simple restrictions on both currents and potential differences. Minimizing the cost of generation is an obvious objective, but there are often others.

Another important class of optimization problems arises in civil engineering. Structures such as buildings and bridges tend to assume positions of minimum constrained potential energy. A typical problem may be to make a structure as light as possible, while being able to satisfy a variety of unforeseen load conditions such as high winds or heavy snowfall. Energy minimization is also important in computational physics, chemistry and biology, where for examples molecules naturally adopt minimum energy configurations. And again, problems that are often considered to be continuous are frequently written in variational form and an appropriate discretization minimized.

The problem of selecting a balanced portfolio between a set of possible assets so as to maximize the expected gain, or to minimize risk or both, lies at the heart of computational finance. Indeed, so important is this problem that Markowitz was awarded the Nobel prize for economics for automating its solution.

And finally, mathematicians like to build parameterized models of physical phenomena, and then to try match such models to reality. This "fitting" of data to models occurs everywhere, and the best fit is, of course, an optimization problem. A good example is image reconstruction where a "hazy" picture needs to be refocused automatically to reveal hidden detail.

Thus we hope we have convinced that optimization arises in important and diverse ways. So now on with the job of finding optima.

PART 1

OPTIMALITY CONDITIONS AND WHY THEY ARE IMPORTANT

1.1 Optimization problems

As we have said optimization is concerned with the minimization or maximization of an objective function, say, $f(x)$. Since

$$\text{maximum } f(x) = -\text{ minimum } (-f(x))$$

there is no loss in generality in concentrating here on minimization—throughout, minimization will take place with respect to an n-vector, x, of real unknowns. A bit of terminology here: the smallest value of f gives its *minimum*, while any (there may be more than one) corresponding values of x are a *minimizer*.

There are a number of important subclasses of optimization problems. The simplest is *unconstrained minimization*, where we aim to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

where the *objective function* $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$. One level up is *equality constrained minimization*, where now we try to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \ \text{ subject to } \ c(x) = 0$$

where the *constraints* $c\colon \mathbb{R}^n \longrightarrow \mathbb{R}^m$. For consistency we shall assume that $m \leq n$, for otherwise it is unlikely (but not impossible) that there is an x that satisfies all of the equality constraints. Another important problem is *inequality constrained minimization*, in which we aim to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \ \text{ subject to } \ c(x) \geq 0$$

where $c\colon \mathbb{R}^n \longrightarrow \mathbb{R}^m$ and now m may be larger than n. The most general problem involves both equality and inequality constraints—some inequalities may have upper as well as lower bounds—and may be further sub-classified depending on the nature of the constraints. For instance, some of the $c_i(x)$ may be linear (that is $c_i(x) = a_i^T x - b_i$ for some vector a_i and scalar b_i), some may be simple bounds on individual components of x (for example, $c_i(x) = x_i$), or some may result from a network ("flow in = flow out").

1.2 Notation

It is convenient to introduce our most common notation and terminology at the outset. Suppose that $f(x)$ is at least twice continuously differentiable ($f \in \mathcal{C}^2$). We let $\nabla_x f(x)$ denote the vector of first partial derivatives, whose i-th component is $\partial f(x)/\partial x_i$. Similarly, the i,j-th component of the (symmetric) matrix $\nabla_{xx} f(x)$ is the second partial derivative $\partial^2 f(x)/\partial x_i \partial x_j$. We also write the usual Euclidean inner product between two p-vectors u and v as $\langle u, v \rangle \stackrel{\text{def}}{=} \sum_{i=1}^p u_i v_i = u^T v$ (and mention, for those who care, that some but not all of what we have to say remains true in more general Hilbert spaces!). We denote the set of points for which all the constraints are satisfied as $\mathcal{C}$, and say that any $x \in \mathcal{C}$ (resp. $x \notin \mathcal{C}$) is *feasible* (resp. *infeasible*).

With this in mind we define the *gradient* and *Hessian* (matrix) of the objective function f to be $g(x) \stackrel{\text{def}}{=} \nabla_x f(x)$ and $H(x) \stackrel{\text{def}}{=} \nabla_{xx} f(x)$, respectively. Likewise, the gradient and Hessian of the i-th constraint are $a_i(x) \stackrel{\text{def}}{=} \nabla_x c_i(x)$ and $H_i(x) \stackrel{\text{def}}{=} \nabla_{xx} c_i(x)$. The *Jacobian* (matrix) is

$$A(x) \stackrel{\text{def}}{=} \nabla_x c(x) \equiv \begin{pmatrix} a_1^T(x) \\ \cdots \\ a_m^T(x) \end{pmatrix}.$$

Finally, if y is a vector (of so-called *Lagrange multipliers*), the *Lagrangian* (function) is

$$\ell(x, y) \stackrel{\text{def}}{=} f(x) - \langle y, c(x) \rangle, \tag{1.1}$$

while its gradient and Hessian with respect to x are, respectively,

$$g(x, y) \stackrel{\text{def}}{=} \nabla_x \ell(x, y) \equiv g(x) - \sum_{i \in \mathcal{M}} y_i a_i(x) \equiv g(x) - A^T(x) y \text{ and} \tag{1.2}$$

$$H(x, y) \stackrel{\text{def}}{=} \nabla_{xx} \ell(x, y) \equiv H(x) - \sum_{i \in \mathcal{M}} y_i H_i(x), \tag{1.3}$$

where $\mathcal{M} = \{1, \ldots, m\}$.

One last piece of notation: e_i is the i-th unit vector, while e is the vector of ones, and I is the (appropriately dimensioned) identity matrix.

1.3 Lipschitz continuity and Taylor's theorem

It might be argued that those who understand Taylor's theorem and have a basic grasp of linear algebra have all the tools they need to study continuous optimization—of course, this leaves aside all the beautiful mathematics needed to fully appreciate optimization in abstract settings.

Taylor's theorem(s) can most easily be stated for functions with Lipschitz continuous derivatives. Let $\mathcal{X}$ and $\mathcal{Y}$ open sets, let $F : \mathcal{X} \to \mathcal{Y}$, and let $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$ be norms on $\mathcal{X}$ and $\mathcal{Y}$ respectively. Then F is *Lipschitz continuous at* $x \in \mathcal{X}$ if there exists a function $\gamma(x)$ such that

$$\|F(z) - F(x)\|_{\mathcal{Y}} \leq \gamma(x) \|z - x\|_{\mathcal{X}}$$

for all $z \in \mathcal{X}$. Moreover F is *Lipschitz continuous throughout/in* $\mathcal{X}$ if there exists a constant γ such that

$$\|F(z) - F(x)\|_{\mathcal{Y}} \leq \gamma \|z - x\|_{\mathcal{X}}$$

for all x and $z \in \mathcal{X}$. Lipschitz continuity relates (either locally or globally) the changes that occur in F to those that are permitted in x.

Armed with this, we have the following *Taylor* approximation results. The first suggests how good (or bad) a first-order (linear) or second-order (quadratic) Taylor series approximation to a scalar-valued function may be.

Theorem 1.1. Let $\mathcal{S}$ be an open subset of $\mathbb{R}^n$, and suppose $f : \mathcal{S} \to \mathbb{R}$ is continuously differentiable throughout $\mathcal{S}$. Suppose further that $g(x)$ is Lipschitz continuous at x, with Lipschitz constant $\gamma^L(x)$ in some appropriate vector norm. Then, if the segment $x + \theta s \in \mathcal{S}$ for all $\theta \in [0, 1]$,

$$|f(x+s) - m^L(x+s)| \le \tfrac{1}{2}\gamma^L(x)\|s\|^2, \text{ where}$$

$$m^L(x+s) = f(x) + \langle g(x), s\rangle.$$

If f is twice continuously differentiable throughout $\mathcal{S}$ and $H(x)$ is Lipschitz continuous at x, with Lipschitz constant $\gamma^Q(x)$,

$$|f(x+s) - m^Q(x+s)| \le \tfrac{1}{6}\gamma^Q(x)\|s\|^3, \text{ where}$$

$$m^Q(x+s) = f(x) + \langle g(x), s\rangle + \tfrac{1}{2}\langle s, H(x)s\rangle.$$

The second result is a variation on the theme of the first, and is often refereed to as the *generalized mean-value* theorem.

Theorem 1.2. Let $\mathcal{S}$ be an open subset of $\mathbb{R}^n$, and suppose $f : \mathcal{S} \to \mathbb{R}$ is twice continuously differentiable throughout $\mathcal{S}$. Suppose further that $s \neq 0$, and that the interval $[x, x+s] \in \mathcal{S}$. Then

$$f(x+s) = f(x) + \langle g(x), s\rangle + \tfrac{1}{2}\langle s, H(z)s\rangle$$

for some $z \in (x, x+s)$.

The third result compares how bad a first-order Taylor series approximation to a vector valued function might be.

Theorem 1.3. Let $\mathcal{S}$ be an open subset of $\mathbb{R}^n$, and suppose $F : \mathcal{S} \to \mathbb{R}^m$ is continuously differentiable throughout $\mathcal{S}$. Suppose further that $\nabla_x F(x)$ is Lipschitz continuous at x, with Lipschitz constant $\gamma^L(x)$ in some appropriate vector norm and its induced matrix norm. Then, if the segment $x + \theta s \in \mathcal{S}$ for all $\theta \in [0, 1]$,

$$\|F(x+s) - M^L(x+s)\| \le \tfrac{1}{2}\gamma^L(x)\|s\|^2,$$

where

$$M^L(x+s) = F(x) + \nabla_x F(x)s$$

1.4 Farkas' lemma — the fundamental theorem of linear inequalities

Most readers probably feel comfortable with systems of linear equations, but are less familiar with linear inequalities. There is a rich theory of the latter, but only one famous result concerns us here. This is the so-called fundamental theorem of linear inequalities, Farkas' lemma, which is the basis for proofs of optimality results for linear and nonlinear optimization.

To set the scene, let $\mathcal{A}$ be a given index set, and suppose $\{a_i\}_{i\in\mathcal{A}}$ are given vectors. The (polyhedral) cone

$$C = \left\{ \sum_{i\in\mathcal{A}} y_i a_i \mid y_i \geq 0 \right\}$$

is the closed[1], convex set of all positive linear combinations of our given vectors. The question we ask is, when does another given vector g lie in $\mathcal{C}$? The answer is, if and only if g is not separated from the vectors $\{a_i\}_{i\in\mathcal{A}}$ by a hyperplane $\langle s, v\rangle = 0$ for some given s (see Figure 1.1).

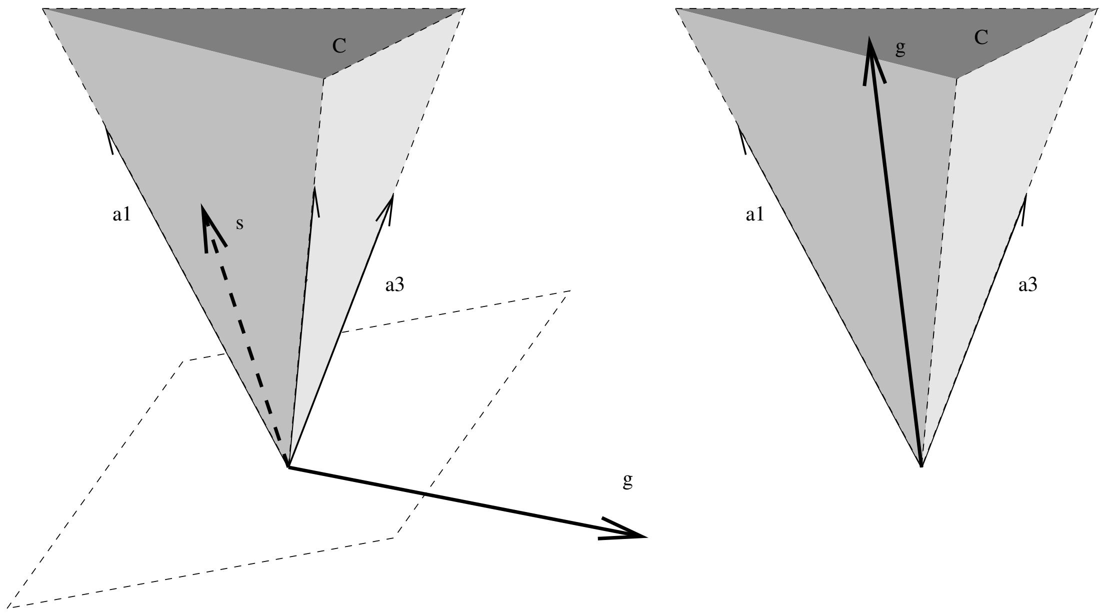

Figure 1.1: In the left-hand illustration, g is not in $\mathcal{C}$, and is separated from $\{a_i\}_{i\in\mathcal{A}}$ by the hyperplane $\langle s, v\rangle = 0$. In the right-hand one, g is a member of $\mathcal{C}$ and not separated from $\mathcal{C}$'s support vectors.

Formally, we have

[1]The fact that $\mathcal{C}$ is closed seems obvious but needs some proof. See e.g., J. Nocedal and S. Wright, "Numerical Optimization", Springer Verlag (1999), p357.

Farkas' lemma. Given any vectors g and a_i, $i \in \mathcal{A}$, the set

$$\mathcal{S} = \{s \mid \langle g, s\rangle < 0 \text{ and } \langle a_i, s\rangle \geq 0 \text{ for } i \in \mathcal{A}\}$$

is empty if and only if

$$g \in C = \left\{\sum_{i\in\mathcal{A}} y_i a_i \mid y_i \geq 0 \text{ for all } i \in \mathcal{A}\right\}.$$

An immediate application is to linear programming. If we are interested in minimizing the linear objective $\langle g, x\rangle$ subject to the linear constraints $\langle a_i, x\rangle \geq b_i$ for $i \in \mathcal{I}$, suppose that we have found a feasible point x, and wish to know if there is a direction s for which we can improve the objective. Then if $\mathcal{A}$ gives the indices of those constraints that are active, $\mathcal{S}$ gives us the set of (locally) improving directions, and Farkas' lemma tells us that improvement will be possible if and only if g is not in $\mathcal{C}$. The reader will most likely recall that $g \in \mathcal{C}$ along with feasibility of x are the optimality conditions for linear programming, and it should not then be surprising that Farkas' lemma has an important role to play in the nonlinear case.

1.5 Optimality conditions

Now is the time to come clean. It is very, very difficult to say anything about the solutions to the optimization problems given in Part 1.1. This is almost entirely because we are considering very general problems, for which there may be many local, often non-global, minimizers. There are two possible ways around this. We might choose to restrict the class of problems we allow, so that all local minimizers are global. But since this would rule out the vast majority of nonlinear problems that arise in practice, we instead choose to lower our sights, and only aim for local minimizers—there are methods that offer some guarantee of global optimality, but to date they are really restricted to small or very specially structured problems.

Formally, we still need to define what we mean by a local minimizer. A feasible point x_* is a *local* minimizer of $f(x)$ if there is an open neighbourhood $\mathcal{N}$ of x_* such that $f(x_*) \leq f(x)$ for all $x \in \mathcal{C} \bigcap \mathcal{N}$. If there is an open neighbourhood $\mathcal{N}$ of x_* such that $f(x_*) < f(x)$ for all $x \neq x_* \in \mathcal{C} \bigcap \mathcal{N}$, it is *isolated*.

While such definitions agree with our intuition, they are of very little use in themselves. What we really need are optimality conditions. Optimality conditions are useful for three reasons. Firstly, the provide a means of guaranteeing that a candidate solution is indeed (locally) optimal—these are the so-called *sufficient conditions*. Secondly, they indicate when a point is not optimal—these are the *necessary conditions*. Finally they guide us in the design of algorithms, since lack of optimality indicates when we may improve our objective. We now give details.

1.6 Optimality conditions for unconstrained minimization

We first consider what we might deduce if we were fortunate enough to have found a local minimizer of $f(x)$. The following two results provide first- and second-order necessary optimality conditions (respectively).

Theorem 1.4. Suppose that $f \in C^1$, and that x_* is a local minimizer of $f(x)$. Then

$$g(x_*) = 0.$$

Theorem 1.5. Suppose that $f \in C^2$, and that x_* is a local minimizer of $f(x)$. Then $g(x_*) = 0$ and $H(x_*)$ is positive semi-definite, that is

$$\langle s, H(x_*)s \rangle \geq 0 \text{ for all } s \in \mathbb{R}^n.$$

But what if we have found a point that satisfies the above conditions? Is it a local minimizer? Yes, an isolated one, provided the following second-order sufficient optimality conditions are satisfied.

Theorem 1.6. Suppose that $f \in C^2$, that x_* satisfies the condition $g(x_*) = 0$, and that additionally $H(x_*)$ is positive definite, that is

$$\langle s, H(x_*)s \rangle > 0 \text{ for all } s \neq 0 \in \mathbb{R}^n.$$

Then x_* is an isolated local minimizer of f.

Notice how slim is the difference between these necessary and sufficient conditions.

1.7 Optimality conditions for constrained minimization

When constraints are present, things get more complicated. In particular, the geometry of the feasible region at (or near) to a minimizer plays a very subtle role. Consider a suspected minimizer x_*. We shall say that a constraint is *active* at x_* if and only if $c_i(x_*) = 0$. By necessity, equality constraints will be active, while determining which (if any) of the inequalities is active is probably the overriding concern in constrained optimization.

In order to say anything about optimality, it is unfortunately necessary to rule out "nasty" local minimizers such as cusps on the constraint boundary. This requires that we have to ask that so-called *constraint qualifications* hold—essentially these say that linear approximations to the constraints characterize all feasible perturbations about x_* and that perturbations which keep strongly active constraints strongly active (a *strongly* active constraint is one that will still be active if the data, and hence minimizer, is slightly perturbed) are completely characterized by their corresponding linearizations being forced to be active. Fortunately, such assumptions are automatically satisfied if the constraints are linear, or if the constraints that are active have independent gradients, and may actually be guaranteed in far weaker circumstances than these.

1.7.1 Optimality conditions for equality-constrained minimization

Given constraint qualifications, first- and second-order necessary optimality conditions for problems involving equality constraints are (respectively) as follows.

Theorem 1.7. Suppose that $f,\ c \in C^1$, and that x_* is a local minimizer of $f(x)$ subject to $c(x) = 0$. Then, so long as a first-order constraint qualification holds, there exist a vector of Lagrange multipliers y_* such that

$$\begin{aligned} c(x_*) &= 0 \quad (\textit{primal feasibility}) \text{ and} \\ g(x_*) - A^T(x_*)y_* &= 0 \quad (\textit{dual feasibility}). \end{aligned}$$

Theorem 1.8. Suppose that $f,\ c \in C^2$, and that x_* is a local minimizer of $f(x)$ subject to $c(x) = 0$. Then, provided that first- and second-order constraint qualifications hold, there exist a vector of Lagrange multipliers y_* such that

$$\langle s, H(x_*, y_*)s\rangle \geq 0 \ \text{ for all } \ s \in \mathcal{N} \tag{1.4}$$

where

$$\mathcal{N} = \{s \in \mathbb{R}^n \mid A(x_*)s = 0\}.$$

Notice that there are two first-order optimality requirements: primal feasibility (the constraints are satisfied), and dual feasibility (the gradient of the objective function is expressible as a linear combination of the gradients of the constraints). It is not hard to anticipate that, just as in the unconstrained case, sufficient conditions occur when the requirement (1.4) is strengthened to $\langle s, H(x_*, y_*)s\rangle > 0$ for all $s \in \mathcal{N}$.

1.7.2 Optimality conditions for inequality-constrained minimization

Finally, when the problem involves inequality constraints, it is easy to imagine that only the constraints that are active at x_* play a role—the inactive constraints play no part in defining the minimizer—and indeed this is so. First- and second-order necessary optimality conditions are (respectively) as follows.

Theorem 1.9. Suppose that $f,\ c \in C^1$, and that x_* is a local minimizer of $f(x)$ subject to $c(x) \geq 0$. Then, provided that a first-order constraint qualification holds, there exist a vector of Lagrange multipliers y_* such that

$$\begin{aligned} c(x_*) &\geq 0 \quad (\textit{primal feasibility}), \\ g(x_*) - A^T(x_*)y_* = 0 \ \text{ and } \ y_* &\geq 0 \quad (\textit{dual feasibility}) \text{ and} \\ c_i(x_*)[y_*]_i &= 0 \quad (\textit{complementary slackness}). \end{aligned} \tag{1.5}$$

Theorem 1.10. Suppose that $f,\ c \in C^2$, and that x_* is a local minimizer of $f(x)$ subject to $c(x) \geq 0$. Then, provided that first- and second-order constraint qualifications hold, there exist a vector of Lagrange multipliers y_* for which primal/dual feasibility and complementary slackness requirements hold as well as

$$\langle s, H(x_*, y_*)s \rangle \geq 0 \ \text{ for all } \ s \in \mathcal{N}_+$$

where

$$\mathcal{N}_+ = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle s, a_i(x_*) \rangle = 0 \ \text{ for all } \ i \in \mathcal{M} \ \text{ such that } \ c_i(x_*) = 0 \ \text{ and } \ [y_*]_i > 0 \text{ and} \\ \langle s, a_i(x_*) \rangle \geq 0 \ \text{ for all } \ i \in \mathcal{M} \ \text{ such that } \ c_i(x_*) = 0 \ \text{ and } \ [y_*]_i = 0 \end{array} \right\}. \tag{1.6}$$

See how dual feasibility now imposes an extra requirement, that the Lagrange multipliers be non-negative—this is where Farkas' lemma comes into play—while as expected there is an additional (complementary slackness) assumption that inactive constraints necessarily have zero Lagrange multipliers. Also notice that $\mathcal{N}_+$, the set over which the Hessian of the Lagrangian is required to be positive semi-definite, may now be the intersection of a linear manifold and a cone, a particularly unpleasant set to work with. The requirements (1.5) are often know as the *Karush-Kuhn Tucker*, or simply *KKT*, conditions in honour of their discoverers.

The by-now obvious sufficient conditions also hold:

Theorem 1.11. Suppose that $f,\ c \in C^2$, and that x_* and a vector of Lagrange multipliers y_* satisfy (1.5) and

$$\langle s, H(x_*, y_*)s \rangle > 0$$

for all s in the set $\mathcal{N}_+$ given in (1.6). Then x_* is an isolated local minimizer of $f(x)$ subject to $c(x) \geq 0$.

PART 2

LINESEARCH METHODS FOR UNCONSTRAINED OPTIMIZATION

In this and the next parts, we shall concentrate on the unconstrained minimization problem,

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x),$$

where the objective function $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$. We shall assume that $f \in C^1$ (sometimes C^2) with Lipschitz continuous derivatives. Often in practice this assumption is violated, but nonetheless the methods converge (perhaps by good fortune) regardless.

Despite knowing how to characterise local minimizers of our problem, in practice it is rather unusual for us to be able to provide or compute an explicit minimizer. Instead, we would normally expect to fall back on a suitable iterative process. An *iteration* is simply a procedure whereby a sequence of points

$$\{x_k\}, \quad k = 1, 2, \ldots$$

is generated, starting from some initial "guess" x_0, with the overall aim of ensuring that (a subsequence) of the $\{x_k\}$ has favourable limiting properties. These might include that any limit generated satisfies first-order or, even better, second-order necessary optimality conditions.

Notice that we will not be able to guarantee that our iteration will converge to a global minimizer unless we know that f obeys very strong conditions, nor regrettably in general that any limit point is even a local minimizer (unless by chance it happens to satisfy second-order sufficiency conditions). What we normally do try to ensure is that, at the very least, the iteration is *globally* convergent, that is that (for at least) a subsequence of iterates $\{g(x_k)\}$ converges to zero. And our hope is that such a sequence converges at a reasonably fast asymptotic rate. These two preoccupations lie at the heart of computational optimization.

For brevity, in what follows, we shall write $f_k = f(x_k)$, $g_k = g(x_k)$ and $H_k = H(x_k)$.

2.1 Linesearch methods

Generically, *linesearch methods* work as follows. Firstly, a *search direction* p_k is calculated from x_k. This direction is required to be a *descent direction*, i.e.,

$$\langle p_k, g_k \rangle < 0 \;\text{ if }\; g_k \neq 0,$$

so that, for small steps along p_k, Taylor's theorem (Theorem 1.1) guarantees that the objective function may be reduced. Secondly, a suitable *steplength* $\alpha_k > 0$ is calculated so that

$$f(x_k + \alpha_k p_k) < f_k.$$

The computation of α_k is the *linesearch*, and may itself be an iteration. Finally, given both search direction and steplength, the iteration concludes by setting

$$x_{k+1} = x_k + \alpha_k p_k.$$

Such a scheme sounds both natural and simple. But as with most simple ideas, it needs to be refined somewhat in order to become a viable technique. What might go wrong? Firstly, consider the example in Figure 2.1.

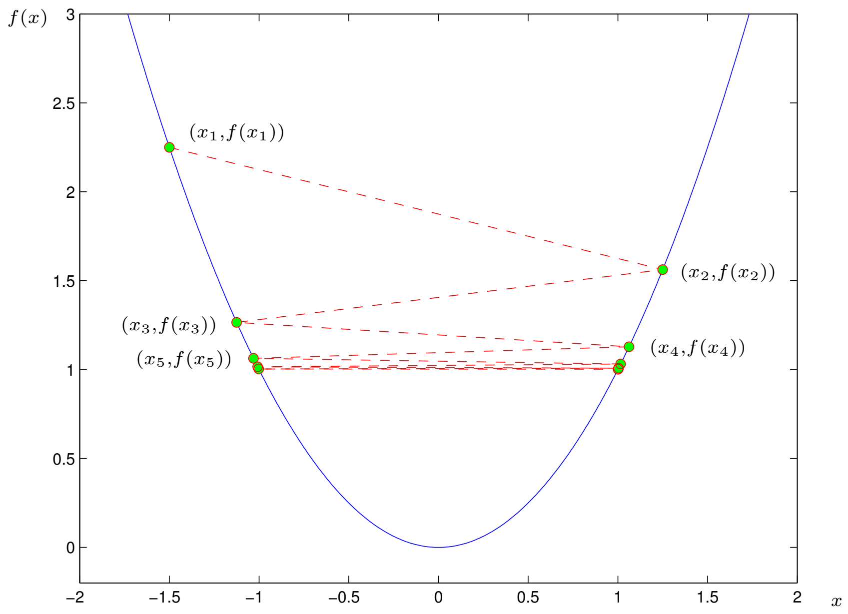

Figure 2.1: The objective function $f(x) = x^2$ and the iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = (-1)^{k+1}$ and steps $\alpha_k = 2 + 3/2^{k+1}$ from $x_0 = 2$.

Here the search direction gives a descent direction, and the iterates oscillate from one side of the minimizer to the other. Unfortunately, the decrease per iteration is ultimately so small that the iterates converge to the pair ± 1, neither of which is a stationary point. What has gone wrong? Simply the steps are too long relative to the amount of objective-function decrease that they provide.

Is this the only kind of failure? Unfortunately, no. For consider the example in Figure 2.2. Now the iterates approach the minimizer from one side, but the stepsizes are so small that each iterate falls woefully short of the minimizer, and ultimately converge to the non-stationary value 1.

So now we can see that a simple-minded linesearch method can fail if the linesearch allows steps that are either too long or too short relative to the amount of decrease that might be obtained with a well-chosen step.

2.2 Practical linesearch methods

In the early days, it was often suggested that α_k should be chosen to minimize $f(x_k + \alpha p_k)$. This is known as an *exact* linesearch. In most cases, exact linesearches prove to be both very expensive—they are essentially univariate minimizations—and most definitely not cost effective, and are consequently rarely used nowadays.

Modern linesearch methods prefer to use *inexact* linesearches, which are guaranteed to pick steps that are neither too long nor too short. In addition, they aim to pick a "useful" initial "guess" for each stepsize so as to ensure fast asymptotic convergence—we will return to this when we discuss Newton's method. The main contenders amongst the many possible inexact linesearches are the

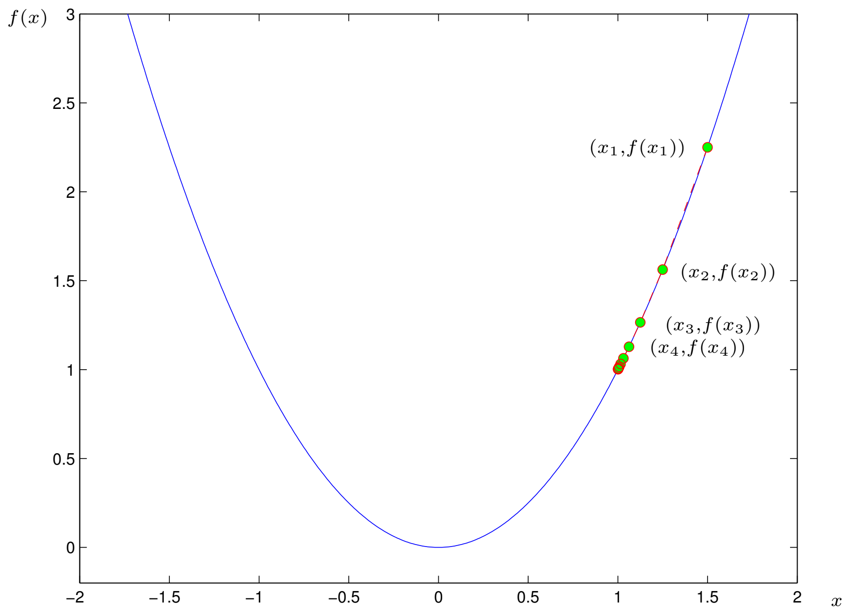

Figure 2.2: The objective function $f(x) = x^2$ and the iterates $x_{k+1} = x_k + \alpha_k p_k$ generated by the descent directions $p_k = -1$ and steps $\alpha_k = 1/2^{k+1}$ from $x_0 = 2$.

so-called "backtracking- Armijo" and the "Armijo-Goldstein" varieties. The former are extremely easy to implement, and form the backbone of most Newton-like linesearch methods. The latter are particularly important when using secant quasi-Newton methods (see Part 2.5.3), but alas we do not have space to describe them here.

Here is a basic *backtracking* linesearch to find α_k:

> Given $\alpha_{\text{init}} > 0$ (e.g., $\alpha_{\text{init}} = 1$),
> let $\alpha^{(0)} = \alpha_{\text{init}}$ and $l = 0$.
> Until $f(x_k + \alpha^{(l)} p_k) < f_k$
>   set $\alpha^{(l+1)} = \tau\alpha^{(l)}$, where $\tau \in (0, 1)$ (e.g., $\tau = \frac{1}{2}$)
>   and increase l by 1.
> Set $\alpha_k = \alpha^{(l)}$.

Notice that the backtracking strategy prevents the step from getting too small, since the first allowable value stepsize of the form $\alpha_{\text{init}}\tau^i$, $i = 0, 1, \ldots$ is accepted. However, as it stands, there is still no mechanism for preventing too large steps relative to decrease in f. What is needed is a tighter requirement than simply that $f(x_k + \alpha^{(l)} p_k) < f_k$. Such a role is played by the Armijo condition.

The *Armijo condition* is that the steplength be asked to give slightly more than simply decrease in f. The actual requirement is that

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \beta \langle p_k, g_k \rangle$$

for some $\beta \in (0,1)$ (e.g., $\beta = 0.1$ or even $\beta = 0.0001$)—this requirement is often said to give *sufficient decrease*. Observe that, since $\langle p_k, g_k \rangle < 0$, the longer the step, the larger the required decrease in f. The range of permitted values for the stepsize is illustrated in Figure 2.3.

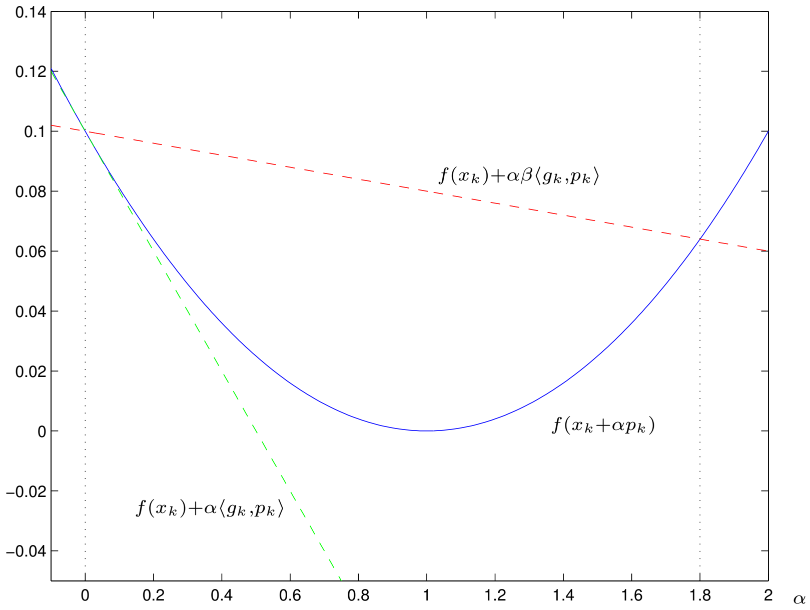

Figure 2.3: A steplength of anything up to 1.8 is permitted for this example, in the case where $\beta = 0.2$.

The Armijo condition may then be inserted into our previous backtracking scheme to give the aptly-named *Backtracking-Armijo* linesearch:

> Given $\alpha_{\text{init}} > 0$ (e.g., $\alpha_{\text{init}} = 1$),
> let $\alpha^{(0)} = \alpha_{\text{init}}$ and $l = 0$.
> Until $f(x_k + \alpha^{(l)} p_k) \leq f(x_k) + \alpha^{(l)} \beta \langle p_k, g_k \rangle$
>   set $\alpha^{(l+1)} = \tau \alpha^{(l)}$, where $\tau \in (0,1)$ (e.g., $\tau = \frac{1}{2}$)
>   and increase l by 1.
> Set $\alpha_k = \alpha^{(l)}$.

Of course, it is one thing to provide likely-sounding rules to control stepsize selection, but another to be sure that they have the desired effect. Indeed, can we even be sure that there are points which satisfy the Armijo condition? Yes, for we have

Theorem 2.1. Suppose that $f \in C^1$, that $g(x)$ is Lipschitz continuous with Lipschitz constant $\gamma(x)$, that $\beta \in (0,1)$ and that p is a descent direction at x. Then the Armijo condition

$$f(x+\alpha p) \leq f(x) + \alpha\beta\langle p, g(x)\rangle$$

is satisfied for all $\alpha \in [0, \alpha_{\max(x,p)}]$, where

$$\alpha_{\max}(x,p) = \frac{2(\beta-1)\langle p, g(x)\rangle}{\gamma(x)\|p\|_2^2}.$$

Note that since $\gamma(x)$ is rarely known, the theorem does not provide a recipe for computing $\alpha_{\max}(x,p)$, merely a guarantee that there is such a suitable value. The numerator in $\alpha_{\max}(x,p)$ corresponds to the slope and the denominator to the curvature term. It can be interpreted as follows: If the curvature term is large, then the admissible range of α is small. Similarly, if the projected gradient along the search direction is large, then the range of admissible α is larger.

It then follows that the Backtracking-Armijo linesearch can be guaranteed to terminate with a suitably modest stepsize.

Corollary 2.2. Suppose that $f \in C^1$, that $g(x)$ is Lipschitz continuous with Lipschitz constant γ_k at x_k, that $\beta \in (0,1)$ and that p_k is a descent direction at x_k. Then the stepsize generated by the backtracking-Armijo linesearch terminates with

$$\alpha_k \geq \min\left(\alpha_{\text{init}}, \frac{2\tau(\beta-1)\langle p_k, g_k\rangle}{\gamma_k\|p_k\|_2^2}\right).$$

Again, since γ_k is rarely known, the corollary does not give a practical means for computing α_k, just an assurance that there is a suitable value. Notice that the stepsize is certainly not too large, since it is bounded above by $\alpha_{\max}$, and can only be small when $\langle p, g(x)\rangle/\|p\|_2^2$ is. This will be the key to the successful termination of generic linesearch methods.

2.3 Convergence of generic linesearch methods

In order to tie all of the above together, we first need to state our Generic Linesearch Method:

> Given an initial guess x_0, let $k = 0$
> Until convergence:
>   Find a descent direction p_k at x_k.
>   Compute a stepsize α_k using a
>   backtracking-Armijo linesearch along p_k.
>   Set $x_{k+1} = x_k + \alpha_k p_k$, and increase k by 1.

It is then quite straightforward to apply Corollary 2.2 to deduce the following very general convergence result.

Theorem 2.3. Suppose that $f \in C^1$ and that g is Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic Linesearch Method,

either

$$g_l = 0 \text{ for some } l \geq 0$$

or

$$\lim_{k \to \infty} f_k = -\infty$$

or

$$\lim_{k \to \infty} \min \left(|\langle p_k, g_k \rangle|, \frac{|\langle p_k, g_k \rangle|}{\|p_k\|_2} \right) = 0.$$

In words, either we find a first-order stationary point in a finite number of iterations, or we encounter a sequence of iterates for which the objective function is unbounded from below, or the slope (or a normalized slope) along the search direction converges to zero. While the first two of these possibilities are straightforward and acceptable consequences, the latter is perhaps not. For one thing, it certainly does not say that the gradient converges to zero, that is the iterates may not ultimately be first-order critical, since it might equally occur if the search direction and gradient tend to be mutually orthogonal. Thus we see that simply requiring that p_k be a descent direction is not a sufficiently demanding requirement. We will return to this shortly, but first we consider *the* archetypical globally convergent algorithm, the method of steepest descent.

2.4 Method of steepest descent

We have just seen that the Generic Linesearch Method may not succeed if the search direction becomes orthogonal to the gradient. Is there a direction for which this is impossible? Yes, when the search direction is the descent direction

$$p_k = -g_k,$$

the so-called *steepest-descent* direction—the epithet is appropriate since this direction solves the problem

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \;\; m_k^L(x_k + p) \stackrel{\text{def}}{=} f_k + \langle p, g_k \rangle \;\text{ subject to }\; \|p\|_2 = \|g_k\|_2,$$

and thus gives the greatest possible reduction in a first-order model of the objective function for a step whose length is specified. Global convergence follows immediately from Theorem 2.3.

Theorem 2.4. Suppose that $f \in C^1$ and that g is Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic Linesearch Method using the steepest-descent direction,

either

$$g_l = 0 \;\text{ for some }\; l \geq 0$$

or

$$\lim_{k \to \infty} f_k = -\infty$$

or

$$\lim_{k \to \infty} g_k = 0.$$

As we mentioned above, this theorem suggests that steepest descent really is the archetypical globally convergent method, and in practice many other methods resort to steepest descent when they run into trouble. However, the method is not scale invariant, as re-scaling variables can lead to widely different "steepest-descent" directions. Even worse, as we can see in Figure 2.4, convergence may be (and actually almost always is) very slow in theory, while numerically convergence sometimes does not occur at all as the iteration stagnates. In practice, steepest-descent is all but worthless in most cases. The figure exhibits quite typical behaviour in which the iterates repeatedly oscillate from one side of a objective function "valley" to the other. All of these phenomena may be attributed to a lack of attention to problem curvature when building the search direction. We now turn to methods that try to avoid this defect.

2.5 More general descent methods

2.5.1 Newton and Newton-like methods

Let B_k be a symmetric, positive definite matrix. Then it is trivial to show that the search direction p_k for which

$$B_k p_k = -g_k$$

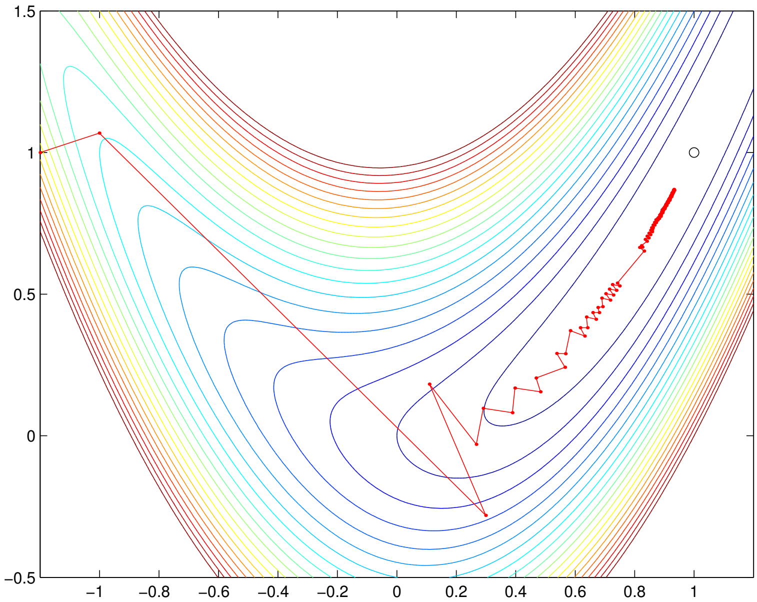

Figure 2.4: Contours for the objective function $f(x, y) = 10(y - x^2)^2 + (x - 1)^2$, and the iterates generated by the Generic Linesearch steepest-descent method.

is a descent direction. In fact, this direction solves the direction-finding problem

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad m_k^Q(x_k + p) \stackrel{\text{def}}{=} f_k + \langle p, g_k \rangle + \tfrac{1}{2}\langle p, B_k p \rangle, \tag{2.1}$$

where $m_k^Q(x_k + p)$ is a quadratic approximation to the objective function at x_k.

Of particular interest is the possibility that $B_k = H_k$, for in this case $m_k^Q(x_k + p)$ gives a second-order Taylor's approximation to $f(x_k + p)$. The resulting direction for which

$$H_k p_k = -g_k$$

is known as the *Newton* direction, and any method which uses it is a Newton method . But notice that the Newton direction is only guaranteed to be useful in a linesearch context if the Hessian H_k is positive definite, for otherwise p_k might turn out to be an ascent direction.

It is also worth saying that while one can motivate such Newton-like methods from the prospective of minimizing a local second-order model of the objective function, one could equally argue that they aim to find a zero of a local first-order model

$$g(x_k + p) \approx g_k + B_k p_k$$

of its gradient. So long as B_k remains "sufficiently" positive definite, we can make precisely the same claims for these second-order methods as for those based on steepest descent.

Theorem 2.5. Suppose that $f \in C^1$ and that g is Lipschitz continuous on $\mathbb{R}^n$. Then, for the iterates generated by the Generic Linesearch Method using the Newton or Newton-like direction,

either

$$g_l = 0 \text{ for some } l \geq 0$$

or

$$\lim_{k\to\infty} f_k = -\infty$$

or

$$\lim_{k\to\infty} g_k = 0$$

provided that the eigenvalues of B_k are uniformly bounded and bounded away from zero.

Indeed, one can regard such methods as "scaled" steepest descent, but they have the advantage that they can be made scale invariant for suitable B_k, and crucially, as we see in Figure 2.5, their

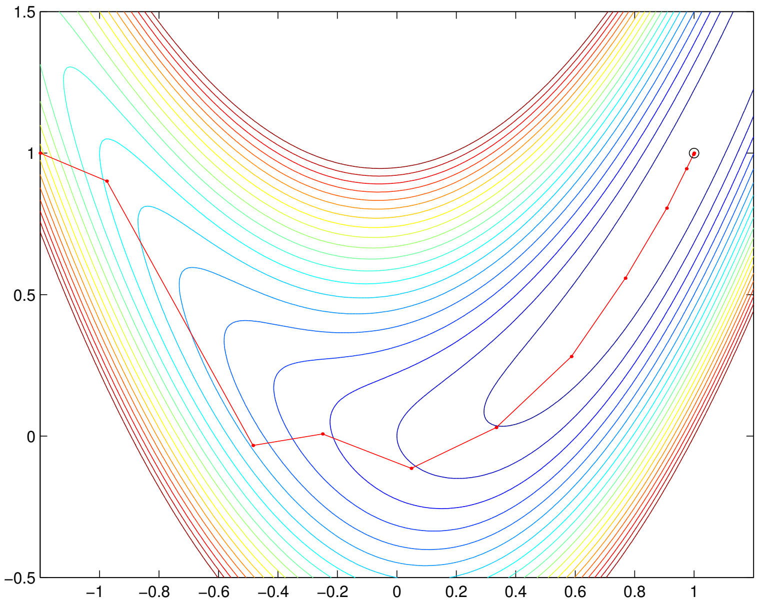

Figure 2.5: Contours for the objective function $f(x, y) = 10(y - x^2)^2 + (x - 1)^2$, and the iterates generated by the Generic Linesearch Newton method.

convergence is often significantly faster than steepest descent. In particular, in the case of the

Newton direction, the Generic Linesearch method will usually converge very rapidly indeed.

Theorem 2.6. Suppose that $f \in C^2$ and that H is Lipschitz continuous on $\mathbb{R}^n$. Then suppose that the iterates generated by the Generic Linesearch Method with $\alpha_{\text{init}} = 1$ and $\beta < \frac{1}{2}$, in which the search direction is chosen to be the Newton direction $p_k = -H_k^{-1} g_k$ whenever H_k is positive definite, has a limit point x_* for which $H(x_*)$ is positive definite. Then

(i) $\alpha_k = 1$ for all sufficiently large k,

(ii) the entire sequence $\{x_k\}$ converges to x_*, and

(iii) the rate is Q-quadratic, i.e, there is a constant $\kappa \geq 0$.

$$\lim_{k \to \infty} \frac{\|x_{k+1} - x_*\|_2}{\|x_k - x_*\|_2^2} \leq \kappa.$$

2.5.2 Modified-Newton methods

Of course, away from a local minimizer there is no reason to believe that H_k will be positive definite, so precautions need to be taken to ensure that Newton and Newton-like linesearch methods, for which B_k is (or is close to) H_k, satisfy the assumptions of the global convergence Theorem 2.5. If H_k is indefinite, it is usual to solve instead

$$(H_k + M_k) p_k \equiv B_k p_k - g_k,$$

where M_k is chosen so that $B_k = H_k + M_k$ is "sufficiently" positive definite and $M_k = 0$ when H_k is itself "sufficiently" positive definite. This may be achieved in a number of ways.

Firstly, if H_k has the spectral (that is eigenvector-eigenvalue) decomposition $H_k = Q_k D_k Q_k^T$, then M_k may be chosen so that

$$B_k \equiv H_k + M_k = Q_k \max(\epsilon I, |D_k|) Q_k^T$$

for some "small" ϵ". This will shift all the insufficiently positive eigenvalues by as little as possible as is needed to make the overall matrix positive definite. While such a decomposition may be too expensive to compute for larger problems, a second, cheaper alternative is to find (or estimate) the smallest (necessarily real!) eigenvalue, $\lambda_{\min}(H_k)$, of H_k, and to set

$$M_k = \max(0, \epsilon - \lambda_{\min}(H_k)) I$$

so as to shift *all* the eigenvalues by just enough as to make the smallest "sufficiently" positive. While this is often tried in practice, in the worst case it may have the effect of over-emphasising one large, negative eigenvalue at the expense of the remaining small, positive ones, and in producing a direction which is essentially steepest descent. Finally, a good compromise is instead to attempt a Cholesky factorization of H_k, and to alter the generated factors if there is evidence that the factorization will otherwise fail—the *Cholesky* factorization of a symmetric, positive-definite matrix is a decomposition into the product of a lower triangular matrix and its transpose. There are a number of so-called *Modified Cholesky* factorizations, each of which will obtain

$$B_k \equiv H_k + M_k = L_k L_k^T,$$

where M_k is zero for sufficiently positive-definite H_k, and "not-unreasonably large" in all other cases.

2.5.3 Quasi-Newton methods

It was fashionable in the 1960s and 1970s to attempts to build suitable approximations B_k to the Hessian, H_k. Activity in this area has subsequently died down, possibly because people started to realize that computing exact second derivatives was not as onerous as they had previously contended, but these techniques are still of interest particularly when gradients are awkward to obtain (such as when the function values are simply given as the result of some other, perhaps hidden, computation). There are broadly two classes of what may be called quasi-Newton methods.

The first are simply based on estimating columns of H_k by *finite differences*. For example, we might use the approximation

$$(H_k)e_i \approx h^{-1}(g(x_k + he_i) - g_k) \stackrel{\text{def}}{=} (B_k)e_i$$

for some "small" scalar $h > 0$. The difficulty here is in choosing an appropriate value for h: too large a value gives inaccurate approximations, while a too small one leads to large numerical cancellation errors.

The second sort of quasi-Newton methods are known as *secant approximations*, and try to ensure the *secant condition*

$$B_{k+1}s_k = y_k, \quad \text{where} \quad s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = g_{k+1} - g_k,$$

that would be true if $H(x)$ were constant, is satisfied. The secant condition gives a lot of flexibility, and among the many methods that have been discovered, the *Symmetric Rank-1* method, for which

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{\langle s_k, y_k - B_k s_k \rangle},$$

and the *BFGS* method, for which

$$B_{k+1} = B_k + \frac{y_k y_k^T}{\langle s_k, y_k \rangle} - \frac{B_k s_k s_k^T B_k}{\langle s_k, B_k s_k \rangle}$$

are the best known (and generally the best). Note that the former may give indefinite approximations (or even fail), while the latter is guaranteed to generate symmetric and positive definite matrices so long as B_0 is positive definite and $\langle s_k, y_k \rangle > 0$ (the last condition may be ensured by an appropriate "Goldstein" linesearch). Since both of these secant methods are based on low-rank updates, it is possible to keep the per-iteration linear algebraic requirements at a more modest level for such methods than is generally possible with Newton or finite-difference methods.

2.5.4 Conjugate-gradient and truncated-Newton methods

And what if the problem is large and matrix factorization is out of the question? We have already considered (and rejected) steepest-descent methods. Is there something between the simplicity of steepest descent and the power (but expense) of Newton-like methods? Fortunately, yes.there is.

Suppose that instead of solving (2.1), we instead find our search direction as

$$p_k = \text{(approximate)} \underset{p \in \mathbb{R}^n}{\arg\min}\; q(p) = f_k + \langle p, g_k \rangle + \tfrac{1}{2}\langle p, B_k p \rangle, \tag{2.2}$$

where we assume that B_k is positive definite—the key word here is *approximate*. Suppose that instead of minimizing q over all $p \in \mathbb{R}^n$, we restrict p to lie in a (much) smaller subspace—of course if we do this we will not (likely) obtain the optimal value of q, but we might hope to obtain a good approximation with considerably less effort.

Let $D^i = (d^0 : \cdots : d^{i-1})$ be any collection of i vectors, let

$$\mathcal{D}^i = \{p \mid p = D^i p_d \;\text{ for some }\; p_d \in \mathbb{R}^i\}$$

be the subspace spanned by D^i, and suppose that we choose to pick

$$p^i = \underset{p \in \mathcal{D}^i}{\arg\min}\; q(p).$$

Then immediately $D^{i\,T} g^i = 0$, where $g^i = B_k p^i + g_k$ is the gradient of q at p^i. More revealingly, since $p^{i-1} \in \mathcal{D}^i$, it follows that $p^i = p^{i-1} + D^i p_d^i$, where

$$\begin{aligned} p_d^i &= \underset{p_d \in \mathbb{R}^i}{\arg\min}\; \langle p_d, D^{i\,T} g^{i-1} \rangle + \tfrac{1}{2}\langle p_d, D^{i\,T} B_k D^i p_d \rangle \\ &= -(D^{i\,T} B_k D^i)^{-1} D^{i\,T} g^{i-1} = -\langle d^{i-1}, g^{i-1} \rangle (D^{i\,T} B_k D^i)^{-1} e_i. \end{aligned}$$

Hence

$$p^i = p^{i-1} - \langle d^{i-1}, g^{i-1} \rangle D^i (D^{i\,T} B_k D^i)^{-1} e_i. \tag{2.3}$$

All of this is true regardless of D^i. But now suppose that the members of $\mathcal{D}^i$ are *B_k-conjugate*, that is to say that $\langle d_i, B_k d_j \rangle = 0$ for all $i \neq j$. If this is so (2.3) becomes

$$p^i = p^{i-1} + \alpha^{i-1} d^{i-1}, \;\text{ where }\; \alpha^{i-1} = -\frac{\langle d^{i-1}, g^{i-1} \rangle}{\langle d^{i-1}, B_k d^{i-1} \rangle}. \tag{2.4}$$

Thus so long as we can generate B_k-conjugate vectors, we can build up successively improving approximations to the minimize of q by solving a sequence of *one-dimensional* minimization problems—the relationship (2.4) may be interpreted as finding α^{i-1} to minimize $q(p^{i-1} + \alpha d^{i-1})$. But can we find suitable B_k-conjugate vectors?

Surprisingly perhaps, yes, it is easy. Since $D^{i\,T} g^i = 0$ and thus $g^i \notin \mathcal{D}^i$, let

$$d^i = -g^i + \sum_{j=0}^{i-1} \beta^{ij} d^j$$

for some unknown β^{ij}. Then elementary manipulation (and a cool head, see p. 134) shows that if we choose β^{ij} so that d^i is B-conjugate to $\mathcal{D}^i$, we obtain the wonderful result that

$$\beta^{ij} = 0 \;\text{ for }\; j < i-1, \text{ and }\; \beta^{i\,i-1} \equiv \beta^{i-1} = \frac{\|g_i\|_2^2}{\|g_{i-1}\|_2^2}.$$

That is, almost all of the β^{ij} are zero! Summing all of this up, we arrive at the method of *conjugate gradients* (CG):

> Given $p^0 = 0$, set $g^0 = g_k$, $d^0 = -g_k$ and $i = 0$.
> Until g^i is "small", iterate:
> $\alpha^i = \|g^i\|_2^2 / \langle d^i, Bd^i \rangle$
> $p^{i+1} = p^i + \alpha^i d^i$
> $g^{i+1} = g^i + \alpha^i B_k d^i$
> $\beta^i = \|g^{i+1}\|_2^2 / \|g^i\|_2^2$
> $d^{i+1} = -g^{i+1} + \beta^i d^i$
> and increase i by 1.

Important features are that $\langle d^j, g^{i+1} \rangle = 0$ and $\langle g^j, g^{i+1} \rangle = 0$ for all $j = 0, \ldots, i$, and most particularly that $\langle p^i, g_k \rangle \leq \langle p^{i-1}, g_k \rangle < 0$ for $i = 1, \ldots, n$, from which we see that *any* $p_k = p^i$ is a descent direction.

In practice the above conjugate gradient iteration may be seen to offer a compromise between the steepest-descent direction (stopping when $i = 1$) and a Newton (-like) direction (stopping when $i = n$). For this reason, using such a curtailed conjugate gradient step within a linesearch (or trust-region) framework is often known as a *truncated*-Newton method. Frequently the size of g^i relative to g_k is used as a stopping criteria, a particularly popular rule being to stop the conjugate-gradient iteration when

$$\|g^i\| \leq \min(\|g_k\|^\omega, \eta)\|g_k\|,$$

where η and $\omega \in (0, 1)$, since then a faster-than-linear asymptotic convergence rate may be achieved if $B_k = H_k$.

The reader may well have already encountered the CG method as a good way of (approximately) solving symmetric, positive linear systems $Bx = b$. Since as we have already noted that the solution to (2.2) necessarily satisfies $B_k p_k = -g_k$ if the symmetric B_k is positive definite, we merely need to assign B_k as B and $-g_k$ to b and apply the CG method to recover increasingly accurate estimates p_k of x. We shall return to this equivalence in Part 4.

2.5.5 Nonlinear conjugate-gradient methods

Although we have only mentioned CG methods in the context of generating search directions, they may actually be applied (with care) more generally. To see how, suppose $f(x)$ is quadratic and that we express $x = x_0 + p$ as a perturbation p around a given point x_0. Then a Taylor expansion gives

$$f(x) = f(x_0 + p) = f(x_0) + \langle p, g(x_0) \rangle + \tfrac{1}{2} \langle p, H(x_0) p \rangle,$$

which we can minimize as a function of p using the CG method. Significantly, if we write $x_i = x_0 + p_i$, this gives $g^i = g(x_0) + H(x_0) p_i = g(x_i)$, while it is worth noting that $\alpha^i = \arg\min f(x_i + \alpha d^i)$.

Thus, we may rewrite the CG algorithm as follows:

> Given x^0 and $g(x_0)$, set $d^0 = -g(x_0)$ and $i = 0$.
> Until $g(x_k)$ "small" iterate
>
> $\alpha^i = \arg\min_{\alpha} f(x_i + \alpha d^i)$
>
> $x_{i+1} = x_i + \alpha^i d^i$
>
> $\beta^i = \|g(x_{i+1})\|_2^2 / \|g(x_i)\|_2^2$
>
> $d^{i+1} = -g(x_{i+1}) + \beta^i d^i$
>
> and increase i by 1

But now, there is no mention of the Hessian of $f(x)$. Indeed, there is no explicit mention that this method was derived for quadratic functions at all. Variants of this method have been applied to general non-quadratic f by replacing the calculation of α^i by a suitable linesearch and by modifying β^i (as necessary) to ensure that d^i is a descent direction at x_i. The principal advantage of such nonlinear conjugate gradient methods is that they only requires function and gradient values, and not Hessians In practice, periodic restarts in the steepest-descent direction prove to be beneficial and sometimes essential.

PART 3

TRUST-REGION METHODS FOR UNCONSTRAINED OPTIMIZATION

In this part, we continue to concentrate on the unconstrained minimization problem, and shall as before assume that the objective function is C^1 (sometimes C^2) with Lipschitz continuous derivatives.

3.1 Linesearch vs. trust-region methods

One might view linesearch methods as naturally "optimistic". Fairly arbitrary search directions are permitted—essentially 50% of all possible directions give descent from a given point—while unruly behaviour is held in check via the linesearch. There is, however, another possibility, that more control is taken when choosing the search direction, with the hope that this will then lead to a higher probability that the (full) step really is useful for reducing the objective. This naturally "conservative" approach is the basis of trust-region methods.

As we have seen, linesearch methods pick a descent direction p_k, then pick a stepsize α_k to "reduce" $f(x_k + \alpha p_k)$ and finally accept $x_{k+1} = x_k + \alpha_k p_k$. *Trust-region* methods, by contrast, pick the overall step s_k to reduce a "model" of $f(x_k + s)$, and accept $x_{k+1} = x_k + s_k$ if the decrease predicted by the model is realised by $f(x_k + s_k)$. Since there is no guarantee that this will always be so, the fall-back mechanism is to set $x_{k+1} = x_k$, and to "refine" the model when the existing model produces a poor step. Thus, while a linesearch method recovers from a poor step by retreating along a parametric (usually linear) curve, a trust-region method recovers by reconsidering the whole step-finding procedure.

3.2 Trust-region models

It is natural to build a model of $f(x_k+s)$ by considering Taylor series approximations. Of particular interest are the *linear* model

$$m_k^L(s) = f_k + \langle s, g_k \rangle,$$

and the *quadratic* model

$$m_k^Q(s) = f_k + \langle s, g_k \rangle + \tfrac{1}{2}\langle s, B_k s \rangle,$$

where B_k is a symmetric approximation to the local Hessian matrix H_k. However, such models are far from perfect. In particular, the models are unlikely to resemble $f(x_k + s)$ if s is large. More seriously, the models may themselves be unbounded from below so that any attempts to minimize them may result in a large step. This defect will always occur for the linear model (unless $g_k = 0$), and also for the quadratic model if B_k is indefinite (and possibly if B_k is only positive semi-definite). Thus simply using a Taylor-series model is fraught with danger.

There is, fortunately, a simple and effective way around this conundrum. The idea is to prevent the model $m_k(s)$ from being unboundedness by imposing a *trust-region* constraint

$$\|s\| \leq \Delta_k,$$

for some "suitable" scalar *radius* $\Delta_k > 0$, on the step. This is a natural idea, since we know from Theorem 1.1 that we can improve the approximation error $|f(x_k + s) - m_k(s)|$ by restricting the allowable step. Thus our *trust-region subproblem* is to

$$\text{approximately} \quad \underset{s \in \mathbb{R}^n}{\text{minimize}} \;\; m_k(s) \;\; \text{subject to} \;\; \|s\| \leq \Delta_k,$$

and we shall choose s_k as approximate solution of this problem. In theory, it does not depend on which norm $\|\cdot\|$ we use (at least, in finite-dimensional spaces), but in practice it might!

For simplicity, we shall concentrate on the second-order (Newton-like) model

$$m_k(s) = m_k^Q(s) = f_k + \langle s, g_k \rangle + \tfrac{1}{2} \langle s, B_k s \rangle$$

and any (consistent) trust-region norm $\|\cdot\|$ for which

$$\kappa_s \|\cdot\| \leq \|\cdot\|_2 \leq \kappa_l \|\cdot\|$$

for some $\kappa_l \geq \kappa_s > 0$. Notice that the gradient of $m_k(s)$ at $s = 0$ coincides with the gradient of f at x_k, and also, unlike for linesearch methods, $B_k = H_k$ is always allowed. The vast majority of models use the ℓ_1, ℓ_2 or ℓ_∞ norms on $\mathbb{R}^n$, and for these we have $\|\cdot\|_2 \leq \|\cdot\|_2 \leq \|\cdot\|_2$ (obviously!!), $n^{-\frac{1}{2}} \|\cdot\|_1 \leq \|\cdot\|_2 \leq \|\cdot\|_1$ and $\|\cdot\|_\infty \leq \|\cdot\|_2 \leq n \|\cdot\|_\infty$.

3.3 Basic trust-region method

Having decided upon a suitable model, we now turn to the trust-region algorithm itself. As we have suggested, we shall choose to "accept" $x_{k+1} = x_k + s_k$ whenever (a reasonable fraction of) the predicted model decrease $f_k - m_k(s_k)$ is realized by the actual decrease $f_k - f(x_k + s_k)$. We measure this by computing the ratio

$$\rho_k = \frac{f_k - f(x_k + s_k)}{f_k - m_k(s_k)}$$

of actual to predicted decrease, and accepting the trust-region step when ρ_k is not unacceptably smaller than 1.0. If the ratio is close to (or larger than) 1.0, there is good reason to believe that future step computations may well benefit from an increase in the trust-region radius, so we allow a radius increase in this case. If, by contrast, there is poor agreement between the actual and predicted decrease (and particularly, if f actually increases), the current step is poor and should be rejected. In this case, we reduce the trust-region radius to encourage a more suitable step at the next iteration.

We may summarize the basic trust-region method as follows:

Given $k = 0$, $\Delta_0 > 0$ and x_0, until "convergence" do:
Build the second-order model $m(s)$ of $f(x_k + s)$.
"Solve" the trust-region subproblem to find s_k
for which $m(s_k)$ "<" f_k and $\|s_k\| \leq \Delta_k$, and define

$$\rho_k = \frac{f_k - f(x_k + s_k)}{f_k - m_k(s_k)}.$$

If $\rho_k \geq \eta_v$ [*very successful*] $\boxed{0 < \eta_v < 1}$
set $x_{k+1} = x_k + s_k$ and $\Delta_{k+1} = \gamma_i \Delta_k$. $\boxed{\gamma_i \geq 1}$
Otherwise if $\rho_k \geq \eta_s$ then [*successful*] $\boxed{0 < \eta_s \leq \eta_v < 1}$
set $x_{k+1} = x_k + s_k$ and $\Delta_{k+1} = \Delta_k$.
Otherwise [*unsuccessful*]
set $x_{k+1} = x_k$ and $\Delta_{k+1} = \gamma_d \Delta_k$. $\boxed{0 < \gamma_d < 1}$
Increase k by 1.

Reasonable values might be $\eta_v = 0.9$ or 0.99, $\eta_s = 0.1$ or 0.01, $\gamma_i = 2$, and $\gamma_d = 0.5$. In practice, these parameters might even be allowed to vary (within reasonable limits) from iteration to iteration. In particular, there would seem to be little justification in increasing the trust region radius following a very successful iteration unless $\|s_k\| \approx \Delta_k$, nor in decreasing the radius by less than is required to "cut off" an unsuccessful s_k.

In practice, the trust-region radius is *not* increased for a very successful iterations, if the step is much shorter, say less than half the trust-region radius. There are various schemes for choosing an initial trust-region radius. However, if the problem is well scaled, then $\Delta_0 = O(1)$ is reasonable. Poor scaling can affect the performance of trust-region methods. In practice it often suffices that the variables of the (scaled) problem have roughly the same order of magnitude.

It remains for us to decide what we mean by "solving" the trust-region subproblem. We shall see in Part 3.5 that (at least in the ℓ_2-trust-region norm case) it is possible to find the (global) solution to the subproblem. However, since this may result in a considerable amount of work, we first seek "minimal" conditions under which we can guarantee convergence of the above algorithm to a first-order critical point.

We have already seen that steepest-descent linesearch methods have very powerful (theoretical) convergence properties. The same is true in the trust-region framework. Formally, at the very least, we shall require that we achieve as much reduction in the model as we would from an iteration of steepest descent. That is, if we define the *Cauchy* point as $s_k^{\mathrm{C}} = -\alpha_k^{\mathrm{C}} g_k$, where

$$\begin{aligned} \alpha_k^{\mathrm{C}} &= \underset{\alpha > 0}{\arg\min}\; m_k(-\alpha g_k) \;\text{ subject to }\; \alpha\|g_k\| \leq \Delta_k \\ &= \underset{0 < \alpha \leq \Delta_k/\|g_k\|}{\arg\min}\; m_k(-\alpha g_k) \end{aligned},$$

we shall require that our step s_k satisfies

$$m_k(s_k) \le m_k(s_k^{\mathrm{C}}) \text{ and } \|s_k\| \le \Delta_k. \tag{3.1}$$

Notice that the Cauchy point is extremely easy to find, since it merely requires that we minimize the quadratic model along a line segment. In practice, we shall hope to—and can—do far better than this, but for now (3.1) suffices.

Figure 3.1 illustrates the trust-region problem in four different situations. The contours of the original function are shown as dotted lines, while the contours of the trust-region model appear as solid lines with the ℓ_2 trust-region ball in bold. Clockwise from top left, the plots depict the following situations: first, a quadratic model with positive definite Hessian, next a linear model about the same point, the third plot shows a quadratic model with indefinite Hessian and the final plot is a quadratic model with positive definite Hessian whose minimizers lies outside the trust-region.

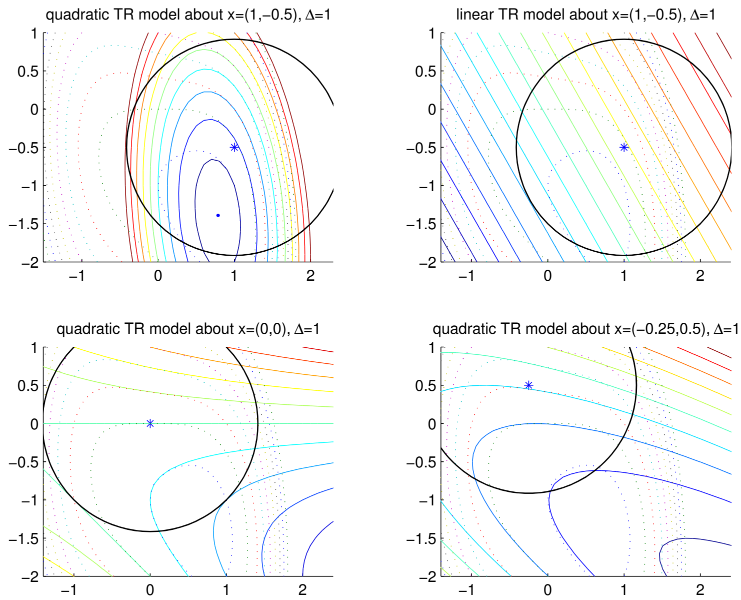

Figure 3.1: Trust-region models of $f(x) = x_1^4 + x_1x_2 + (1+x_2)^2$ about different points.

We now examine the convergence of this trust-region method.

3.4 Basic convergence of trust-region methods

The first thing to note is that we can guarantee a reasonable reduction in the model at the Cauchy point.

Theorem 3.1. If $m_k(s)$ is the second-order model and s_k^{C} is its Cauchy point within the trust-region $\|s\| \leq \Delta_k$, then

$$f_k - m_k(s_k^{\mathrm{C}}) \geq \tfrac{1}{2}\|g_k\|_2 \min\left[\frac{\|g_k\|_2}{1 + \|B_k\|_2}, \kappa_s \Delta_k\right].$$

Observe that the guaranteed reduction depends on how large the current gradient is, and is also affected by the size of both the trust-region radius and the (inverse) of the Hessian.

Since our algorithm requires that the step does at least as well as the Cauchy point, we then have the following immediate corollary.

Corollary 3.2. If $m_k(s)$ is the second-order model, and s_k is an improvement on the Cauchy point within the trust-region $\|s\| \leq \Delta_k$,

$$f_k - m_k(s_k) \geq \tfrac{1}{2}\|g_k\|_2 \min\left[\frac{\|g_k\|_2}{1 + \|B_k\|_2}, \kappa_s \Delta_k\right].$$

This is a typical trust-region result, in that it relates the model reduction to a measure of the distance to optimality, in this case measured in terms of the norm of the gradient.

It is also necessary to say something about how much the model and the objective can vary. Since we are using a second-order model for which the first-two terms are exactly those from the Taylor's approximation, it is not difficult to believe that the difference between model and function will vary like the square of the norm of s_k, and indeed this is so.

Lemma 3.3. Suppose that $f \in C^2$, and that the true and model Hessians satisfy the bounds $\|H(x)\|_2 \leq \kappa_h$ for all x and $\|B_k\|_2 \leq \kappa_b$ for all k and some $\kappa_h \geq 1$ and $\kappa_b \geq 0$. Then

$$|f(x_k + s_k) - m_k(s_k)| \leq \kappa_d \Delta_k^2,$$

where $\kappa_d = \frac{1}{2}\kappa_l^2(\kappa_h + \kappa_b)$, for all k.

Actually the result is slightly weaker than necessary since, for our purposes, we have chosen to replace $\|s_k\|$ by its (trust-region) bound Δ_k. Moreover, rather than requiring a uniform bound on $H(x)$, all that is actually needed is a similar bound for all x between x_k and $x_k + s_k$.

Armed with these bounds, we now arrive at a crucial result, namely that it will always be possible to make progress from a non-optimal point ($g_k \neq 0$).

Lemma 3.4. Suppose that $f \in C^2$, that the true and model Hessians satisfy the bounds $\|H_k\|_2 \leq \kappa_h$ and $\|B_k\|_2 \leq \kappa_b$ for all k and some $\kappa_h \geq 1$ and $\kappa_b \geq 0$, and that $\kappa_d = \frac{1}{2}\kappa_l^2(\kappa_h + \kappa_b)$. Suppose furthermore that $g_k \neq 0$ and that

$$\Delta_k \leq \|g_k\|_2 \min\left(\frac{1}{\kappa_s(\kappa_h + \kappa_b)}, \frac{\kappa_s(1-\eta_v)}{2\kappa_d}\right).$$

Then iteration k is very successful and

$$\Delta_{k+1} \geq \Delta_k.$$

This result is fairly intuitive, since when the radius shrinks the model looks more and more like its first-order Taylor's expansion (provided B_k is bounded) and thus ultimately there must be good local agreement between the model and objective functions.

The next result is a variation on its predecessor, and says that the radius is uniformly bounded away from zero if the same is true of the sequence of gradients, that is the radius will not shrink to zero at non-optimal points.

Lemma 3.5. Suppose that $f \in C^2$, that the true and model Hessians satisfy the bounds $\|H_k\|_2 \leq \kappa_h$ and $\|B_k\|_2 \leq \kappa_b$ for all k and some $\kappa_h \geq 1$ and $\kappa_b \geq 0$, and that $\kappa_d = \frac{1}{2}\kappa_l^2(\kappa_h + \kappa_b)$. Suppose furthermore that there exists a constant $\epsilon > 0$ such that $\|g_k\|_2 \geq \epsilon$ for all k. Then

$$\Delta_k \geq \kappa_\epsilon \stackrel{\text{def}}{=} \epsilon\gamma_d \min\left(\frac{1}{\kappa_s(\kappa_h + \kappa_b)}, \frac{\kappa_s(1-\eta_v)}{2\kappa_d}\right)$$

for all k.

We may then deduce that if there are only a finite number of successful iterations, the iterates must be first-order optimal after the last of these.

Lemma 3.6. Suppose that $f \in C^2$, and that both the true and model Hessians remain bounded for all k. Suppose furthermore that there are only finitely many successful iterations. Then $x_k = x_*$ for all sufficiently large k and $g(x_*) = 0$.

Having ruled out this special (and highly unlikely) case, we then have our first global convergence result, namely that otherwise (and so long as our objective is bounded from below) there is at least one sequence of gradients that converge to zero.

Theorem 3.7. Suppose that $f \in C^2$, and that both the true and model Hessians remain bounded for all k. Then either

$$g_l = 0 \text{ for some } l \geq 0$$

or

$$\lim_{k \to \infty} f_k = -\infty$$

or

$$\liminf_{k \to \infty} \|g_k\| = 0.$$

Is this all we can show? Is it possible for a second sub-sequence of gradients to stay bounded away from zero? Fortunately, no.

Corollary 3.8. Suppose that $f \in C^2$, and that both the true and model Hessians remain bounded for all k. Then either

$$g_l = 0 \text{ for some } l \geq 0$$

or

$$\lim_{k \to \infty} f_k = -\infty$$

or

$$\lim_{k \to \infty} g_k = 0.$$

Thus we have the highly-satisfying result that the gradients of the sequence $\{x_k\}$ generated by our algorithm converge to, or are all ultimately, zero so longs as f is bounded below. This does not mean that a subsequence of $\{x_k\}$ itself converges, but if it does, the limit is first-order critical.

It is also possible to show that an enhanced version of our basic algorithm converges to points satisfying second-order necessary optimality conditions. To do so, we need to ensure that the Hessian of the model converges to that of the objective (as would obviously be the case if $B_k = H_k$), and that the step s_k has a significant component along the eigenvector corresponding to the most negative eigenvalue of B_k (if any). It is also possible to show that if $B_k = H_k$, if $\{x_k\}$ has a limit x_* for which $H(x_*)$ is positive definite, and if s_k is chosen to

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \quad m_k(s) \text{ subject to } \|s\| \leq \Delta_k, \tag{3.2}$$

the step Δ_k stays bounded away from zero, and thus the iteration ultimately becomes Newton's method (c.f. (2.1)).

In conclusion, we have seen that trust-region methods have a very rich underlying convergence theory. But so much for theory. We now turn to the outstanding practical issue, namely how one might hope to find a suitable step s_k. We will consider two possibilities, one that aims to get a very good approximation to (3.2), and a second, perhaps less ambitious method that is more geared towards large-scale computation.

3.5 Solving the trust-region subproblem

For brevity, we will temporarily drop the iteration subscript, and consider the problem of

$$\text{(approximately)} \underset{s \in \mathbb{R}^n}{\text{minimize}} \quad q(s) \equiv \langle s, g \rangle + \tfrac{1}{2}\langle s, Bs \rangle \text{ subject to } \|s\| \leq \Delta. \tag{3.3}$$

As we have already mentioned, our aim is to find s_* so that

$$q(s_*) \leq q(s^{\mathrm{C}}) \text{ and } \|s_*\| \leq \Delta,$$

where s^{C} is the Cauchy point. We shall consider two approaches in this part. The first aims to solve (3.3) exactly, in which case our trust-region method will be akin to a Newton-like method. The second aims for an approximate solution using a conjugate-gradient like method. For simplicity, we shall only consider the ℓ_2-trust region $\|s\| \leq \Delta$, mainly because there are very powerful methods in this case, but of course other norms are possible and are sometimes preferred in practice.

3.5.1 Solving the ℓ_2-norm trust-region subproblem

There is a really powerful solution characterisation result for the ℓ_2-norm trust-region subproblem.

Theorem 3.9. Any *global* minimizer s_* of $q(s)$ subject to $\|s\|_2 \leq \Delta$ satisfies the equation

$$(B + \lambda_* I)s_* = -g,$$

where $B + \lambda_* I$ is positive semi-definite, $\lambda_* \geq 0$ and $\lambda_*(\|s_*\|_2 - \Delta) = 0$. If $B + \lambda_* I$ is positive definite, s_* is unique.

This result is extraordinary as it is very unusual to be able to give necessary and sufficient *global* optimality conditions for a non-convex optimization problem (that is, a problem which might have a number of local minimizers). Even more extraordinary is the fact that the necessary and sufficient conditions are identical. But most crucially, these optimality conditions also suggest how we might solve the problem.

There are two cases to consider. If B is positive definite and the solution s to

$$Bs = -g \tag{3.4}$$

satisfies $\|s\|_2 \leq \Delta$, then it immediately follows that $s_* = s$ ($\lambda_* = 0$ in Theorem 3.9)—this potential solution may simply be checked by seeing if B has Cholesky factors and, if so, using these factors to solve (3.4) $Bs = -g$ and subsequently evaluate $\|s\|_2$. Otherwise, either B is positive definite but the solution to (3.4) satisfies $\|s\|_2 > \Delta$ or B is singular or indefinite. In these cases, Theorem 3.9 then says that s_* satisfies

$$(B + \lambda I)s = -g \text{ and } \langle s, s\rangle = \Delta^2, \tag{3.5}$$

which is a *nonlinear* (quadratic) system of algebraic equations in the $n+1$ unknowns s and λ. Thus, we now concentrate on methods for solving this system.

Suppose B has the spectral decomposition

$$B = U^T \Lambda U;$$

here U is a matrix of (orthonormal) eigenvectors while the diagonal matrix Λ is made up of eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Theorem 3.9 requires that $B + \lambda I$ be positive semi-definite, and so the solution (s, λ) to (3.5) that we seek necessarily satisfies $\lambda \geq -\lambda_1$. The first part of (3.5) enables us to write s explicitly in terms of λ, that is

$$s(\lambda) = -(B + \lambda I)^{-1} g;$$

we will temporarily disregard the possibility that the theorem permits a singular $B + \lambda I$. Notice that once we have found λ,

$$(B + \lambda I)s = -g \tag{3.6}$$

is a linear system. In this case, we may substitute $s(\lambda)$ into the second part of (3.5) to reveal that

$$\psi(\lambda) \stackrel{\text{def}}{=} \|s(\lambda)\|_2^2 = \|U^T(\Lambda + \lambda I)^{-1} U g\|_2^2 = \sum_{i=1}^{n} \frac{\gamma_i^2}{(\lambda_i + \lambda)^2} = \Delta^2, \tag{3.7}$$

where $\gamma_i = \langle e_i, Ug\rangle = \langle U^T e_i, g\rangle$. Thus to solve the trust-region subproblem, it appears that all we have to do is find a particular root of a univariate nonlinear equation.

We illustrate this in Figures 3.2–3.4.

The first shows a convex example (B positive definite). For Δ^2 larger than roughly 1.15, the solution to the problem lies in the interior of the trust region, and may be found directly from (3.4). When Δ is smaller than this, the solution lies on the boundary of the trust region, and can be found as the right-most root of (3.7). The second example is non-convex (B indefinite). Now the solution must lie on the boundary of the trust region for all values of Δ, and again can be found as the right-most root of (3.7), to the right of $-\lambda_1$.

In both Figures 3.2 and 3.3 everything seems easy, and at least a semblance of an algorithm is obvious. But now consider the example in Figure 3.4. This example is especially chosen so that the coefficient γ_1 in (3.7) is zero, that is g is orthogonal to the eigenvector u_1 of B corresponding to the eigenvalue $\lambda_1 = -2$. Remember that Theorem 3.9 tells us that $\lambda \geq 2 = -\lambda_1$. But Figure 3.4 shows that there is no such root of (3.7) if Δ^2 is larger than (roughly) 0.09.

This is an example of what has become known as the *hard* case, which always arises when $\lambda_1 < 0$, $\langle u_1, g\rangle = 0$ and Δ is too big. What is happening? Quite simply, in the hard case $\lambda = -\lambda_1$ and (3.6) is a singular (but consistent) system—it is consistent precisely because $\langle u_1, g\rangle = 0$. But this system has other solutions $s + \alpha u_1$ for any α, because

$$(B + \lambda I)(s + \alpha u_1) = -g,$$

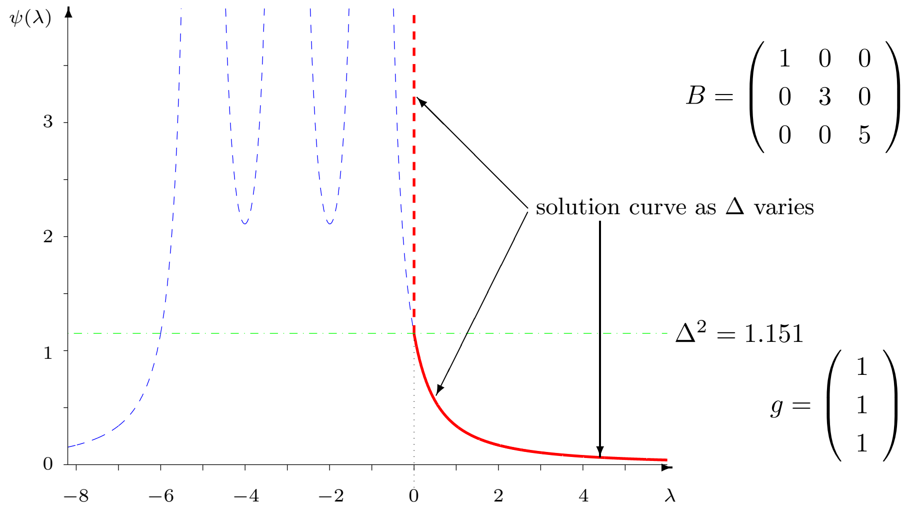

Figure 3.2: A plot of $\psi(\lambda)$ as λ varies from -8 to 6. Note the poles at the negatives of the eigenvalues of H. The heavy curve plots λ against Δ; the dashed vertical component corresponds to interior solutions which occur for all Δ^2 larger than roughly 1.15, while the remaining segment indicates boundary solutions.

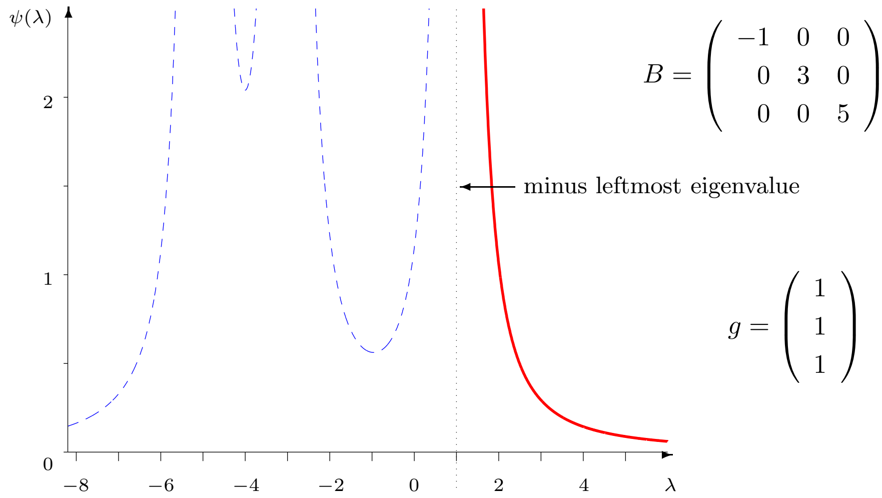

Figure 3.3: A plot of $\psi(\lambda)$ as λ varies from -8 to 6. Again, note the poles at the negatives of the eigenvalues of H.

and u_1 is an eigenvector of $B + \lambda I$. The solution we require is that for which $\|s + \alpha u_1\|_2^2 = \Delta^2$, which is a quadratic equation for the unknown α, and either root suffices.

In the easy (that is not "hard") case, it remains to see how best to solve $|s(\lambda)\|_2 = \Delta$. The answer is blunt. Don't! At least, not directly, since as the previous figures showed, $\psi(\lambda)$ is an unappealing

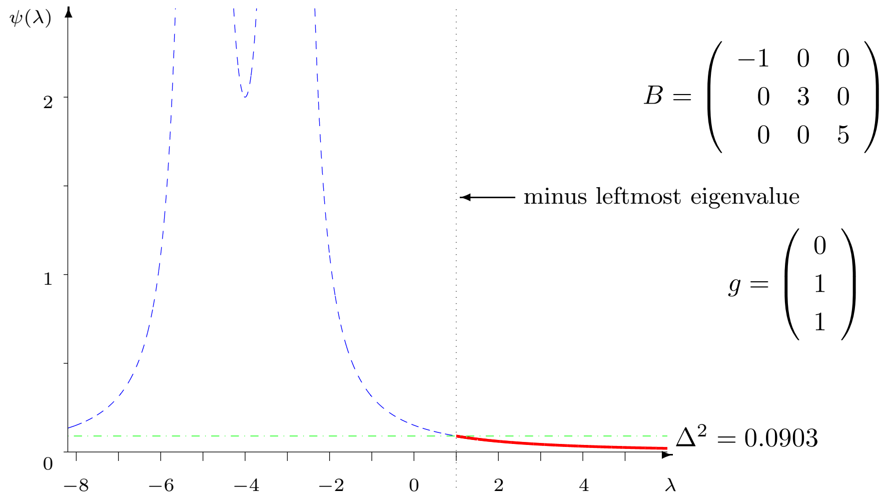

Figure 3.4: A plot of $\psi(\lambda)$ for the modified model as λ varies from -8 to 6. Note that there is no solution with to the equation $\psi(\lambda) = \Delta^2$ with $\lambda \geq 1$ for Δ^2 larger than roughly 0.09.

function with many poles. It is far better to solve the equivalent *secular* equation

$$\phi(\lambda) \stackrel{\text{def}}{=} \frac{1}{\|s(\lambda)\|_2} - \frac{1}{\Delta} = 0,$$

as this has no poles, indeed it is an analytic function, and thus ideal for Newton's method. We illustrate the secular equation in Figure 3.5.

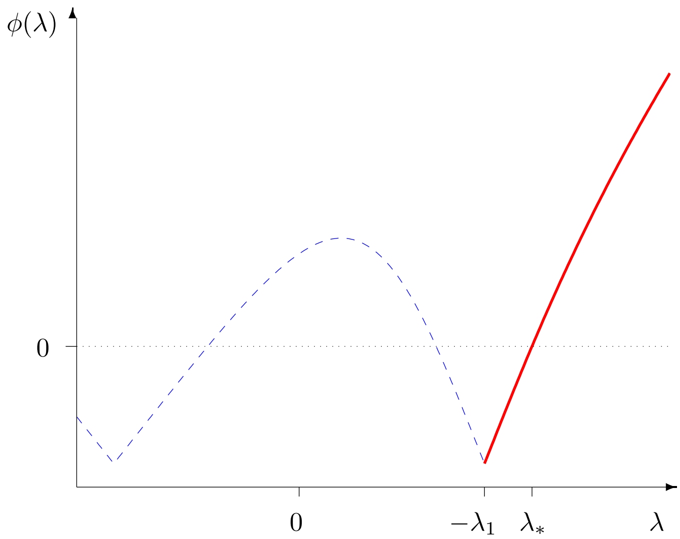

Figure 3.5: A plot of $\phi(\lambda)$ against λ for the problem of minimizing $-\frac{1}{4}s_1^2 + \frac{1}{4}s_2^2 + \frac{1}{2}s_1 + s_2$ subject to $\|s\|_2 \leq 4$.

Without giving details (for these, see the appendix, page 143), Newton's method for the secular equation is as follows

> Let $\lambda > -\lambda_1$ and $\Delta > 0$ be given.
> Until "convergence" do:
>   Factorize $B + \lambda I = LL^T$.
>   Solve $LL^T s = -g$.
>   Solve $Lw = s$.
>   Replace λ by
> $$\lambda + \left(\frac{\|s\|_2 - \Delta}{\Delta}\right)\left(\frac{\|s\|_2^2}{\|w\|_2^2}\right).$$

This is globally and ultimately quadratically convergent when started in the interval $[-\lambda_1, \lambda_*]$ except in the hard case, but needs to be safeguarded to make it robust for the hard and interior solution cases. Notice that the main computational cost per iteration is a Cholesky factorization of $B + \lambda I$, and while this may be reasonable for small problems, it may prove unacceptably expensive when the number of variables is large. We consider an alternative for this case next.

3.6 Solving the large-scale problem

Solving the large-scale trust-region subproblem using the above method is likely out of the question in all but very special cases. The obvious alternative is to use an iterative method to approximate its solution. The simplest approximation that is consistent with our fundamental requirement that we do as least as well as we would at the Cauchy point is to use the Cauchy point itself. Of course, this is simply the steepest descent method, and thus unlikely to be a practical method. The obvious generalization is the conjugate-gradient method, since the first step of CG is in the steepest-descent direction and, as subsequent CG steps further reduce the model, any step generated by the method is allowed by our theory. However, there are a number of other issues we need to address first. In particular, what about the interaction between conjugate gradients and the trust region? And what if B is indefinite?

The conjugate-gradient method to find an approximation to a minimizer of $q(s)$ may be summarised as follows.

Given $s^0 = 0$, set $g^0 = g$, $d^0 = -g$ and $i = 0$.
Until "breakdown" or g^i "small", iterate:

$\alpha^i = \|g^i\|_2^2 / \langle d^i, Bd^i \rangle$
$s^{i+1} = s^i + \alpha^i d^i$
$g^{i+1} = g^i + \alpha^i Bd^i$
$\beta^i = \|g^{i+1}\|_2^2 / \|g^i\|_2^2$
$d^{i+1} = -g^{i+1} + \beta^i d^i$
and increase i by 1.

Notice that we have inserted a termination statement concerning "breakdown". This is intended to cover the fatal case when $\langle d^i, Bd^i \rangle = 0$ (or, in practice, is close to zero), for which the iteration is undefined, and the non-fatal case when $\langle d^i, Bd^i \rangle < 0$ for which $q(s)$ is unbounded from below along the so-called direction of negative curvature d_i.

But what of the trust-region constraint? Here we have a crucial result.

Theorem 3.10. Suppose that the conjugate gradient method is applied to minimize $q(s)$ starting from $s^0 = 0$, and that $\langle d^i, Bd^i \rangle > 0$ for $0 \leq i \leq k$. Then the iterates s^j satisfy the inequalities

$$\|s^j\|_2 < \|s^{j+1}\|_2$$

for $0 \leq j \leq k - 1$.

Simply put, since the norm of the approximate solution generated by the conjugate gradients increases in norm at each iteration, if there is an iteration for which $\|s^j\|_2 > \Delta$, it must be that the solution to the trust-region subproblem lies on the trust-region boundary. That is $\|s_*\|_2 = \Delta$. This then suggests that we should apply the basic conjugate-gradient method above but terminate at iteration i if either (a) $\langle d^i, Bd^i \rangle \leq 0$, since this implies that $q(s)$ is unbounded along d^i, or (b) $\|s^i + \alpha^i d^i\|_2 > \Delta$, since this implies that the solution must lie on the trust-region boundary. In both cases, the simplest strategy is to stop on the boundary at $s = s^i + \alpha^{\mathrm{B}} d^i$, where α^{B} chosen as positive root of the quadratic equation

$$\|s^i + \alpha^{\mathrm{B}} d^i\|_2^2 = \Delta^2.$$

Crucially this s satisfies

$$q(s) \leq q(s^{\mathrm{C}}) \text{ and } \|s\|_2 \leq \Delta$$

and thus Corollary 3.8 shows that the overall trust-region algorithm converges to a first-order critical point.

How good is this truncated conjugate-gradient strategy? In the convex case, it turns out to be very good. Indeed, no worse than half optimal!

Theorem 3.11. Suppose that the truncated conjugate gradient method is applied to approximately minimize $q(s)$ within $\|s\|_2 \leq \Delta$, and that B is positive definite. Then the computed and actual solutions to the problem, s and s_*, satisfy the bound $q(s) \leq \frac{1}{2} q(s_*)$.

In the non-convex (B_k indefinite) case, however, the strategy may be rather poor. For example, if $g = 0$ and B is indefinite, the above truncated conjugate-gradient method will terminate at $s = 0$, while the true solution lies on the trust-region boundary.

What can we do in the non-convex case? The answer is quite involved, but one possibility is to recall that conjugate-gradients is trying to solve the overall problem by successively solving the problem over a sequence of nested subspaces. As we saw, the CG method uses B-conjugate subspaces. But there is an equivalent method, the *Lanczos* method, that uses instead orthonormal bases. Essentially this may be achieved by applying the Gram-Schmidt procedure to the CG basis $\mathcal{D}^i$ to build the equivalent basis $\mathcal{Q}^i = \{s \mid s = Q^i s_q \text{ for some } s_q \in \mathbb{R}^i\}$. It is easy to show that for this Q^i,

$$Q^{i\,T} Q^i = I \text{ and } Q^{i\,T} B Q^i = T^i,$$

where T^i is tridiagonal, and $Q^{i\,T} g = \|g\|_2 \, e_1$, and it is trivial to generate Q^i from the CG $\mathcal{D}^i$. In this case the trust-region subproblem (3.3) may be rewritten as

$$s_q^i = \underset{s_q \in \mathcal{R}^i}{\arg\min} \; \|g\|_2 \, \langle e_1, s_q \rangle + \tfrac{1}{2} \langle s_q, T^i s_q \rangle \text{ subject to } \|s_q\|_2 \leq \Delta,$$

where $s^i = Q^i s_q^i$. Since T^i is tridiagonal, $T^i + \lambda I$ has very sparse Cholesky factors, and thus we can afford to solve this problem using the earlier secular equation approach. Moreover, since we will need to solve a sequence of related problems over nested subspaces, it is easy to imagine that one can use the solution for one problem to initialize the next. In practice, since the approach is equivalent to conjugate gradients, it is best to use CG until the trust-region boundary is reached and then to switch to the Lanczos method at that stage. Such a method has turned out to be most effective in practice.

PART 4

ACTIVE-SET METHODS FOR LINEARLY CONSTRAINED OPTIMIZATION

We now leave the world of unconstrained optimization behind, and move to one involving constraints. We do this gently, by first considering the important class of problems for which the constraints are linear. In its most general form, the *linearly-constrained* optimization problem is to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \;\; \text{subject to} \;\; Ax \left\{ \begin{array}{c} \geq \\ = \end{array} \right\} b, \tag{4.1}$$

where the feasible set is a *polytope*, made up from a finite number of linear equations and/or inequalities. The particularly famous special case of *linear programming* (LP) occurs when $f(x) = \langle g, x \rangle$; there are very special (and very efficient) methods for LP, some of which are based on the observation that for this problem the solution must lie at a vertex of the feasible region. However, LP is perhaps too specialised for a general discussion. More interesting is the *quadratic programming* (QP) case where $f(x) = \langle g, x \rangle + \frac{1}{2} x^T H x$ for some given symmetric H. Almost all of the central ideas for the general problem (4.1) may be seen for QP, and so we need not apologise for devoting the overwhelming part of this partto this special case.

4.1 Quadratic programming

So, we now concentrate on the quadratic programming problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; q(x) = \langle g, x \rangle + \tfrac{1}{2} \langle x, Hx \rangle \;\; \text{subject to} \;\; Ax \geq b \tag{4.2}$$

where H is a given n by n symmetric matrix, g is an n vector,

$$A = \begin{pmatrix} a_1^T \\ \vdots \\ a_m^T \end{pmatrix} \;\; \text{and} \;\; b = \begin{pmatrix} [b]_1 \\ \vdots \\ [b]_m \end{pmatrix}.$$

In practice, the constraints may have both lower and upper bounds, $b^l \leq Ax \leq b^u$, might include equalities, $A^e x = b^e$, may involve simple bounds, $x^l \leq x \leq x^u$, and could have other structurally-significant components, such as those that arise from networks. While good implementations would invariably aim to cope with specially-structured constraints, the essential ideas may be conveyed just by looking at the generic form given in (4.2).

Traditionally quadratic programs are classified according to the properties of their Hessian matrices. A QP is *convex* if H is positive semi-definite (i.e., $\langle x, Hx \rangle \geq 0$ for all x). For such problems, any local minimizer is a global one, and of course linear programming is the special case for which $H = 0$. More specifically, a QP is *strictly convex* if H is positive definite (i.e., $\langle x, Hx \rangle > 0$ for all $x \neq 0$), and for these the minimizer will be unique, if there is a solution at all—it is of course entirely possible that the constraints $Ax \geq b$ have no feasible points; such problems are said to be *infeasible*. The third class of QP are the *non-convex* ones for which H may be indefinite (i.e., $\langle x, Hx \rangle < 0$ for some x). These may have (exponentially) many local minimizers, and indeed the problem may be unbounded from below if the feasible set $\mathcal{C} = \{x \mid Ax \geq b\}$ is unbounded. In any event, for non-convex QPs we will often have to be content with finding a local minimizer. We illustrate both convex and non-convex problems in Figure 4.1

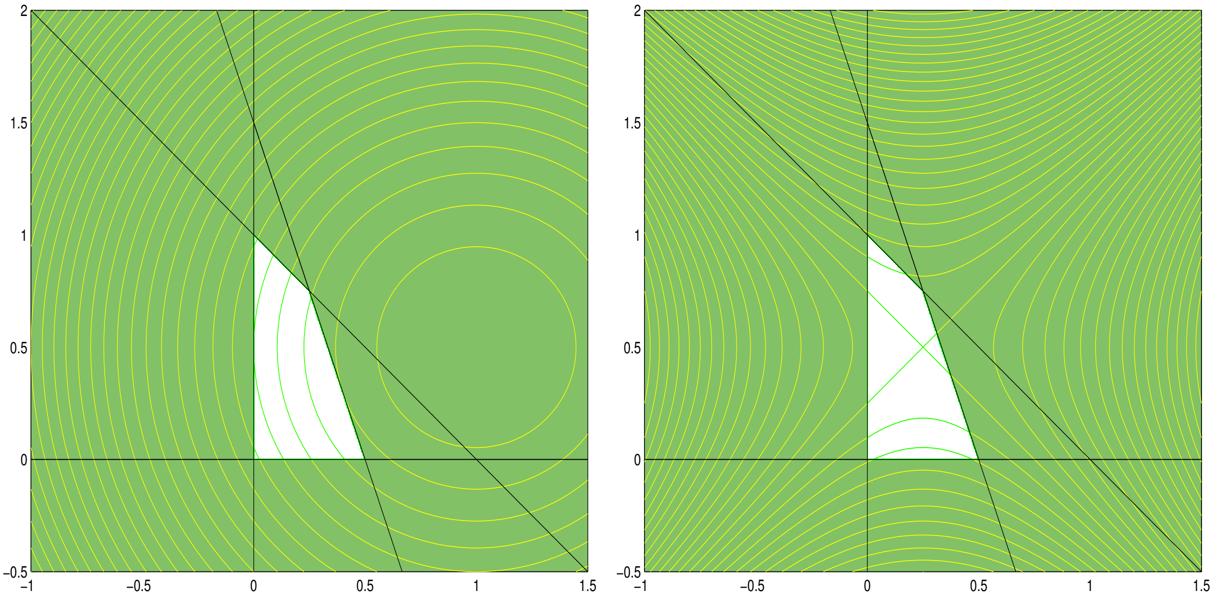

Figure 4.1: Contours of the objective function for (left) a strictly convex problem: $(x_1 - 1)^2 + (x_2 - 0.5)^2$ and (right) a non-convex problem $-2(x_1 - 0.25)^2 + 2(x_2 - 0.5)^2$, and the feasible region for the constraints $x_1 + x_2 \leq 1$, $3x_1 + x_2 \leq 1.5$ and $(x_1, x_2) \geq 0$. Note that the convex problem has a unique minimizer, while the non-convex one has two local minimizers.

Convexity is just one of the issues that affects our ability to solve quadratic programs. Another is size. We say a problem is *small* if the values/structure of the matrix data H and A is irrelevant. Currently problems for which $\min(m,\ n) = O(10^2)$ are small. By contrast, the problem is *large* if exploiting the structure of H and A is important when solving the problem. By today's standards, problems for which $\min(m,\ n) \geq O(10^3)$ might be though of as large. One stage worse, a problem is *huge* if factorizations involving H and A are unrealistic, and currently general problems for which $\min(m,\ n) \geq O(10^5)$ are in this category. Clearly the size is vital, for as we shall see some methods depend crucially on factorizations, while others may (to a certain extent) avoid them.

The reader might wonder if QP is really important, or merely an academic excuse to analyse a rather idealized problem. To this, we might point to the myriad of applications (in, for example, portfolio or structural analysis, VLSI design, discrete-time stabilization, optimal and fuzzy control, finite impulse response design, optimal power flow and economic dispatch; there are currently at least 500 application papers on the subject). Just as importantly, and the reason we gave in the introduction for our interest, they provide in many senses the *prototypical* nonlinear programming problem—-the only essential thing they lack is constraint curvature. And finally, they provide the basic *subproblem* in constrained optimization. For if we are really interested in solving

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) \;\; \text{subject to} \;\; c(x) \geq 0$$

then a simple-minded quadratic approximation to the objective and linearizations of the constraints for perturbations s around x reveal the QP

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \quad f(x) + \langle g(x), s\rangle + \tfrac{1}{2}\langle s, Hs\rangle \;\text{ subject to }\; A(x)s + c(x) \geq 0$$

for some suitable Hessian H. This forms the basis of what are known as sequential quadratic programming (SQP) methods, and we will return to this last point in Part 7.

4.2 Optimality conditions for quadratic programing

Having convinced the reader that QP is a worthwhile problem, we now turn to ways of solving the problem. As always, our first concern must be to examine optimality conditions, both so that we are able to recognise a solution if found by deign or by accident, and more importantly to guide our development of algorithms.

The optimality conditions are simply a direct application of Theorems 1.9–1.11. Any point x_* that satisfies the conditions

$$\begin{aligned} Ax_* &\geq b && \text{(primal feasibility)} \\ Hx_* + g - A^T y_* = 0 \;\text{ and }\; y_* &\geq 0 && \text{(dual feasibility)} \\ [Ax_* - b]_i \cdot [y_*]_i = 0 \;\text{ for all }\; i &\in \mathcal{M} && \text{(complementary slackness)} \end{aligned} \tag{4.3}$$

for some vector of Lagrange multipliers y_* is a *first-order critical* or Karush-Kuhn-Tucker (KKT) point for (4.2); these conditions are necessary for x_* to solve a QP (Theorem 1.9), and notice that no constraint qualification is needed as the constraints are linear. When $[Ax_* - b]_i = 0$ if and only if $[y_*]_i > 0$ for all $1 \leq i \leq m$, the solution is said to be *strictly complementary*. Any first-order critical point x_* for which additionally

$$\langle s, Hs\rangle \geq 0 \;\text{ (respectively} > 0) \;\text{ for all }\; s \in \mathcal{N}_+,$$

where

$$\mathcal{N}_+ = \left\{ s \;\middle|\; \begin{array}{l} \langle a_i, s\rangle = 0 \;\text{ for all }\; i \in \mathcal{M} \;\text{ such that }\; \langle a_i, x_*\rangle = [b]_i \;\text{ and }\; [y_*]_i > 0 \;\text{ and} \\ \langle a_i, s\rangle \geq 0 \;\text{ for all }\; i \in \mathcal{M} \;\text{ such that }\; \langle a_i, x_*\rangle = [b]_i \;\text{ and }\; [y_*]_i = 0 \end{array} \right\},$$

is a *second-order* (respectively *strong second-order*) critical point for (4.2). A solution to QP must be a second-order critical point (Theorem 1.10), while any strong second-order critical point will be an isolated solution to QP (Theorem 1.11). Since (by definition) there are no third (or higher) order terms in QP, it is perhaps not so surprising that the second-order necessary optimality conditions turn out also to be sufficient:

Theorem 4.1. Suppose that $q(x) = \langle g, x\rangle + \frac{1}{2}\langle x, Hx\rangle$. Then x_* is a local minimizer of $q(x)$ within $Ax \geq b$ if and only if x_* is a second-order critical point. It is an isolated local minimizer if and only if x_* is a strong second-order critical point.

As we have already hinted, checking that a point is (strong) second-order critical is very hard because the set $\mathcal{N}_+$ is awkward. With this in mind, we say that any first-order critical point x_* for which additionally

$$\langle s, Hs\rangle \geq 0 \text{ for all } s \in \mathcal{N}$$

where

$$\mathcal{N} = \{s \mid \langle a_i, s\rangle = 0 \text{ for all } i \in \mathcal{M} \text{ such that } \langle a_i, x_*\rangle = [b]_i\},$$

is a *weak* second-order critical point. Although a weak second-order critical point may be a maximizer, checking for weak second-order criticality turns out to be easy, and of course weak and strong criticality coincide if all the active Lagrange multipliers are strictly positive. We illustrate this in Figure 4.2 where H is positive definite along either of the constraints emanating from the origin (a first-order critical point), but not over the set $\mathcal{N}_+$.

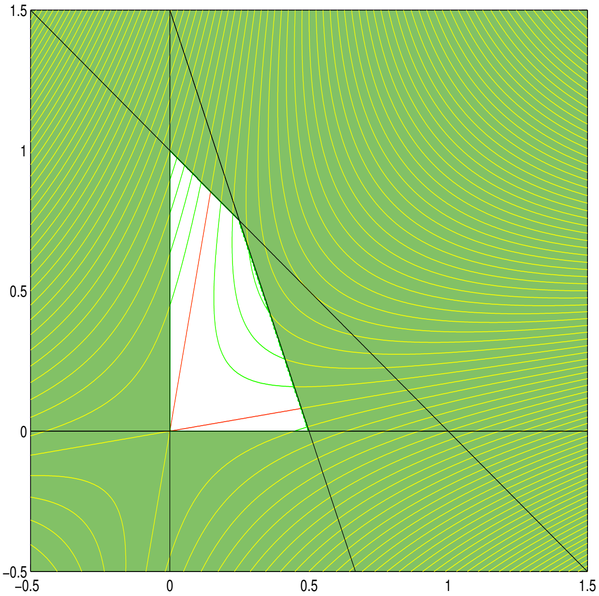

Figure 4.2: Another non-convex problem: contours of the objective function $x_1^2 + x_2^2 - 6x_1x_2$ and the feasible region for the constraints $x_1 + x_2 \leq 1$, $3x_1 + x_2 \leq 1.5$ and $(x_1, x_2) \geq 0$. Note that escaping from the origin may be difficult!

For convex QP, recall that any first-order critical point is a global minimizer. In the strictly convex case, the problem

$$\underset{y \in \mathbb{R}^m,\ y \geq 0}{\text{maximize}} \ -\tfrac{1}{2}\langle g, H^{-1}g\rangle + \langle y, AH^{-1}g + b\rangle - \tfrac{1}{2}\langle y, AH^{-1}A^T y\rangle \tag{4.4}$$

is known as the *dual* of (4.2)—by analogy, (4.2) is the *primal*. The importance of duality is that the primal and dual share optimality conditions. If the primal is feasible, the optimal value of the primal is the same as that of the dual. In some circumstances, such as for example where H (and thus its inverse) are diagonal, it may be more convenient to solve the dual as its constraint set $y \geq 0$ is simpler. The dual (4.4) may be generalized for the convex, as opposed to strictly convex, case, but of course cannot be represented using the inverse of H.

4.3 Algorithms for quadratic programing

We now consider algorithms for solving QP. Essentially there are two classes of methods, although this is actually a slight simplification. The first are known as *active set methods*, and may be subcategorized as *primal* active set methods, which aim to achieve dual feasibility while maintaining primal feasibility and complementary slackness, and *dual* active set methods, which aim to achieve primal feasibility while maintaining dual feasibility and complementary slackness. Primal active-set methods will be the subject of this part. The other class of important algorithms are *interior-point methods*, and these aim to achieve complementary slackness while maintaining primal and dual feasibility. We will examine this kind of method in some detail in Part 6, but note that specially tailored and highly-efficient interior-point methods for QP have been developed.

4.3.1 Equality constrained quadratic programing

The basic subproblem in all of the active-set (and other) methods we will consider is the *equality-constrained* quadratic program (EQP), whose aim is to

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \langle g, x\rangle + \tfrac{1}{2}\langle x, Hx\rangle \ \text{ subject to } \ Ax = 0. \tag{4.5}$$

Notice that, as the name suggests, all the constraints are equalities, and note that we are considering the case where the constraints are homogeneous, that is the right-hand sides are zero—this is all we need for what follows, but is actually easy to consider general constraints $Ax = b$ so long as there is some easy way to find x_0 satisfying $Ax_0 = b$ since then the solution to the general problem is $x + x_0$ provided we add Hx_0 to the linear term g before solving.

We shall assume in what follows that A is of full-rank, but in practice this may mean that we need to pre-process the problem to ensure that this is so. When solving (4.5), there are four possibilities. Firstly, if H is *second-order sufficient* over A, which is to say that $\langle s, Hs\rangle > 0$ for all $s \neq 0$ for which $As = 0$, the unique minimizer to (4.5) satisfies the first-order optimality conditions

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix} \tag{4.6}$$

for some Lagrange multipliers y. Secondly, if x and y satisfy (4.6) and if H is *second-order necessary* over A, which is to say that $\langle s, Hs\rangle \geq 0$ for all s for which $As = 0$, but there is a vector s such that $Hs = 0$ and $As = 0$, then there is a family of *weak* minimizers $x + \alpha s$ for all $\alpha \in \mathbb{R}$. Thirdly, if there is a *direction of linear infinite descent* s for which $As = 0$, $Hs = 0$ and $\langle g, s\rangle < 0$, $q(x + \alpha s)$ is unbounded from below as α increases to infinity. Lastly, if there is *direction of negative curvature* s for which $As = 0$ and $\langle s, Hs\rangle < 0$, $q(x + \alpha s)$ is unbounded from below as α approaches plus or minus infinity. The first of these possibilities is, of course, the only possible outcome if the problem is strict convex, while the last is impossible if the problem is convex.

4.3.1.1 Solving equality-constrained quadratic programs by direct methods

Thus the key to solving (4.5) is really in solving the linear system (4.6) when H is second-order sufficient. The ideal method would be one that can verify the latter while solving the former. We now examine to what extent this is possible. Traditionally there have been three basic approaches—full-space, range-space and null-space—for solving structured indefinite linear systems of the form (4.6), while generic methods for linear systems are usually classified as either direct (factorization) or iterative (e.g., conjugate gradients) based.

The *full-space* (or, as it is sometimes known, KKT or augmented system) approach treats the system matrix

$$K = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \tag{4.7}$$

as a generic symmetric, indefinite matrix, and typically solves (4.6) using a factorization of K. The best-known factorization is of the Bunch-Parlett type, and decomposes $K = PLBL^TP^T$ where P is a permutation, L is unit lower-triangular and B is block diagonal with 1 by 1 and 2 by 2 blocks. This is simply a stable generalization of the Cholesky factorization for indefinite matrices. There are efficient variants (e.g., within LAPACK) for small problems and effectives ones (e.g. SBLKKT, `MA27`/`MA57`, PARDISO or SPOOLES) for large ones. The following result shows that it is easy to check for second-order sufficiency.

Theorem 4.2. Suppose that the m by n matrix A is of full rank. Then H is second-order sufficient over A if and only if the matrix K in (4.7) is non-singular and has precisely m negative eigenvalues.

A strong advantage of the above factorization is that Sylvester's law of inertia tells us that K has B have the same number of negative eigenvalues, and counting these for the latter is trivial.

The *range-space* approach is aimed at problems for which H is (easily) invertible. Simply, x is eliminated from the first block equation in (4.6) and substituted in the second block to determine y from

$$AH^{-1}A^Ty = AH^{-1}g \tag{4.8}$$

followed by x from

$$Hx = -g + A^Ty.$$

This is a typical *Schur complement approach*—the (negative of) $AH^{-1}A^T$ is known as the Schur complement of H in $\mathcal{K}$—in which one solution block of a structured linear system is eliminated to reveal a condensed (Schur complement) system for the remaining solution block. In the strictly convex case, both H and $AH^{-1}A^T$ are positive definite, so that Cholesky factorizations of both are possible. More generally, it is again easy to check for second-order sufficiency.

Theorem 4.3. Suppose that the m by n matrix A is of full rank and the n by n symmetric matrix H is non-singular . Then H is second-order sufficient over A if and only if the matrices H and $AH^{-1}A^T$ have the same number of negative eigenvalues.

Notice that unless H is diagonal, $AH^{-1}A^T$ will almost inevitably be dense, which suggests that a factorization is only appropriate for small m.

By contrast, the *null-space* approach is most appealing when $n - m$ is small. The idea here is simply to note that the second block equation in (4.6) requires that x lies in the null-space of A. So, if S is an n by $n - m$ basis for the null-space of A, and thus $AS = 0$, x may be expressed as $Sx_{_N}$ for some $x_{_N}$. But then the pre-multiplying the first block equation in (4.6) by S^T implies that

$$S^T HSx_{_S} = -S^T g. \tag{4.9}$$

Once again it is easy to check for second-order sufficiency.

Theorem 4.4. Suppose that the m by n matrix A is of full rank and that the columns of S give a basis for the null-space of A. Then H is second-order sufficient over A if and only if the matrix $S^T HS$ is positive definite .

Thus one may anticipate using a Cholesky factorization of the *reduced Hessian* $S^T HS$ when H is second-order sufficient. The main challenge is to find a suitable null-space basis. There are two main approaches. In the first, suppose that we may permute the columns of A so that the first m are linearly independent. Equivalently, suppose that $A = (A_1 \;\; A_2)P$ where P is a permutation matrix and A_1 is non-singular. Then trivially

$$S = P^T \begin{pmatrix} -A_1^{-1}A_2 \\ I \end{pmatrix}$$

gives a *non-orthonormal basis*. This approach is useful for large problems, as the flexibility in selecting A_1 allows choice on sparsity grounds—A_1 will need to be factorized to use S effectively. Since non-orthomal bases may worsen the conditioning of (4.9), our second approach uses an *orthonormal basis* To do this, let

$$A = (L \;\; 0)Q = (L \;\; 0)\begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} = LQ_1$$

be an LQ factorization of A, in which Q is orthonormal and L lower triangular—the reader may well be more familiar with the QR factorization of a matrix, but the LQ factorization of A is simply the QR factorization of A^T. Since then $A(Q_1^T \;\; Q_2^T) = (L \;\; 0)$, immediately $AQ_2^T = 0$, where Q_2 has orthonormal columns, and thus $S = Q_2^T$ gives a possible null-space basis. This is generally a more stable approach, but may be unsuitable for large problems because of the cost of finding the LQ factors. Notice that given x, the required Lagrange multipliers from (4.6) satisfy

$$A_1^T y = P_1(Hx + g),$$

where P_1 are the first m rows of P, with our non-orthonormal basis approach, and

$$L^T y = Q_1(Hx + g)$$

for our orthonormal version.

4.3.1.2 Solving equality-constrained quadratic programs by iterative methods

While matrix factorization is often an excellent way to solve symmetric linear systems of equations $Bx = b$, we have already seen in Section 2.5.4 that an alternative is to use an iterative approach, in which increasingly accurate estimates of the solution are generated as the iteration proceeds. The best-known methods are based on finding approximate solutions from the *Krylov space*

$$\mathcal{K} = \{r^0, Br^0, B(Br^0), B(B(Br^0)), \ldots\},$$

where $r^0 = b - Bx^0$ is the residual at some initial estimate x_0 of the solution. When B is positive definite, this leads to the conjugate-gradient (CG) method, while there is an equally powerful method, MINRES, for the indefinite case. As we saw in the case of unconstrained minimization, it often suffices to find an approximate rather than exact solution to the system under consideration, and usually it pays to accelerate convergence by (explicitly or implicitly) *preconditioning* the system, i.e., to solve

$$C^{-1}Bx = C^{-1}b,$$

where solution with C is easy and ideally $C^{-1}B \approx I$. Notice that the power and flexibility of preconditioned Krylov methods is derived from only requiring matrix-vector products involving B, and solutions involving the *preconditioner* C.

Returning to our three basic approaches for solving (4.6), if H is positive definite and H^{-1} is available (either directly or implicitly via factorization), the range-space approach can simply use the CG method to solve (4.8). Notice that the required matrix-vector product $AH^{-1}A^Td^i = \left(A\left(H^{-1}(A^Td^i)\right)\right)$ may be performed as a sequence of simpler products. The main concern is that any preconditioner will need to approximate the (likely dense) matrix $AH^{-1}A^T$. If H^{-1} not available, it is sometimes possible to use what is known as Urzawa's method, iterating both on solutions to

$$AH^{-1}A^Ty = AH^{-1}g \quad \text{and} \quad Hx = -g + A^Ty$$

at the same time, but convergence is not necessarily guaranteed.

So long as a null-space basis is easy to compute, the null-space equation (4.9) is equally suited to the CG method. Again the matrix vector product $S^THSd_N^i = \left(S^T\left(H(Sd_N^i)\right)\right)$ may be performed as a sequence of simpler products, but again preconditioning can be tricky as there is a need to approximate the likely dense S^THS. The null-space CG method has an additional advantage, namely that if the CG method encounters d_N^i for which $d_N^{i\,T}(S^THS)d_N^i < 0$, then $s = Nd_N^i$ is a direction of negative curvature since $As = 0$ and $\langle s, Hs\rangle < 0$. Moreover, any CG approximation x^i automatically satisfies the constraint $Ax^i = 0$.

Since (4.7) is indefinite, a full-space iterative method would seen condemned to use algorithms like MINRES. Since these methods rely on positive-definite preconditioners, the best hope would be to precondition with a matrix of the form

$$\begin{pmatrix} M & 0 \\ 0 & AN^{-1}A^T \end{pmatrix}$$

where M and N approximate H. Aside from it being far from obvious in general how to choose suitable M and N, MINRES and its rivals also suffer because Ax^i may not be zero for approximate solutions x^i. Although CG may fail if applied to general indefinite problems, in our case it is fortunate that CG is possible provide the preconditioner

$$\begin{pmatrix} M & A^T \\ A & 0 \end{pmatrix}, \tag{4.10}$$

for some second-order sufficient approximation M to H, is used. This may actually be viewed as an implicit null-space approach, and shares the advantage that $Ax^i = 0$ for any preconditioned CG iterate x^i. The main obstacles to this approach are the choice of M and the need to solve with (4.10), but in many cases factorization of (4.10) is possible for a variety of useful M.

4.3.2 Active set algorithms

Armed with the knowledge of how to solve equality-constrained problems, we now return to our main concern, problem (4.2). A fundamental quantity at any point x is the *active set*

$$\mathcal{A}(x) = \{i \mid \langle a_i, x\rangle = [b]_i\}.$$

Most significantly, if x_* solves QP, we have

$$x_* = \arg\min q(x) \text{ subject to } Ax \geq b \equiv \arg\min q(x) \text{ subject to } \langle a_i, x\rangle = [b]_i \text{ for all } i \in \mathcal{A}(x_*),$$

and the remaining, inactive, constraints are irrelevant—by analogy, the *inactive set* is

$$\mathcal{I}(x) = \{i \mid \langle a_i, x\rangle \neq [b]_i\} = \mathcal{M} \setminus \mathcal{A}(x).$$

Just as importantly, a *working set* $\mathcal{W}(x)$ at x is a subset of the active set for which the vectors $\{a_i\}$, $i \in \mathcal{W}(x)$, are linearly independent

The basic idea behind *active set algorithms* is to pick a subset $\mathcal{W}_k$ of $\mathcal{M}$ for which $\{a_i\}$, $i \in \mathcal{W}_k$ are linearly independent, and then to find

$$x_{k+1} = \arg\min q(x) \text{ subject to } \langle a_i, x\rangle = [b]_i \text{ for all } i \in \mathcal{W}_k.$$

If x_{k+1} does not solve (4.2), simply adjust $\mathcal{W}_k$ to form $\mathcal{W}_{k+1}$ and repeat the process—notice that $\mathcal{W}_k = \mathcal{W}(x_{k+1})$ is thus a working set. In principle this is a natural and simple strategy, but the crucial issues are to know when x_{k+1} solves (4.2), and, if it isn't, how to pick the next $\mathcal{W}_{k+1}$. For clarity, we shall let A_k denote the rows of A that are indexed by $\mathcal{W}_k$, and similarly the components of b_k are those of b indexed by $\mathcal{W}_k$.

4.3.2.1 Primal active set algorithms

An important feature of *primal active set algorithms* is that all iterates are required to be (primal) feasible, i.e., $Ax_k \geq b$. Suppose that this is true for iterate x_k, and that the working set $\mathcal{W}_k \subseteq \mathcal{A}(x_k)$. This then implies that $A_k x_k = b_k$ and $A_k x_{k+1} = b_k$, and thus that $x_{k+1} = x_k + s_k$, where s_k solves the equality-constrained QP

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \quad q(x_k + s) \text{ subject to } A_k s = 0, \tag{4.11}$$

a problem we are by now very familiar with. But what if $x_k + s_k$ is not feasible? Then it must be that a currently inactive constraint, say constraint j, becomes active at $x_k + \alpha_k s_k$ for some $\alpha_k < 1$; if more than one becomes active, we pick the smallest such α_k. Then rather than moving to the infeasible $x_k + s_k$, we move instead to the feasible point $x_{k+1} = x_k + \alpha_k s_k$, and *add* constraint j to the working set by assigning $\mathcal{W}_{k+1} = \mathcal{W}_k + \{j\}$. Hence iterate x_{k+1} will be feasible if x_k is, and thus for this to be a workable algorithm, we need to find need an initial feasible point x_0—we will return to this shortly. Notice that $\mathcal{W}_{k+1}$ is a working set because the columns of the full-rank A_k and a_j must be linearly independent since $A_k s_k = 0$ while $\langle a_j, s_k \rangle < 0$. The observant reader will have noticed that if there is an active constraint, say constraint j, that is not in $\mathcal{W}_k$, then $\alpha_k = 0$ if $\langle a_j, s_j \rangle < 0$. Again this needs clarification, and we shall deal with this shortly.

Now suppose that $x_{k+1} = x_k + s_k$ is feasible. There are then three possibilities. Firstly, it may be that $q(x_{k+1}) = -\infty$, which is only possible if the problem is not strictly-convex case—we shall discount such a possibility at this stage. Otherwise,

$$x_{k+1} = \arg\min q(x) \;\text{ subject to }\; \langle a_i, x \rangle = [b]_i \;\text{ for all }\; i \in \mathcal{W}_k$$

is finite, and first-order necessary optimality conditions (Theorem 1.7) give that

$$\begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} x_{k+1} \\ -y_{k+1} \end{pmatrix} = \begin{pmatrix} -g \\ b_k \end{pmatrix}, \tag{4.12}$$

or equivalently

$$\begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} s_k \\ -y_{k+1} \end{pmatrix} = \begin{pmatrix} -g_k \\ 0 \end{pmatrix} \tag{4.13}$$

with $g_k = g + Hx_k$, for some Lagrange multipliers y_{k+1}. The second, happy, outcome is that $y_{k+1} \geq 0$, from which it follows immediately from (4.3) that $(x_*, y_*) = (x_{k+1}, (y_{k+1}, 0))$ is a first-order critical point—here $(y_{k+1}, 0)$ simply means the vector whose components are $[y_{k+1}]_i$ if $i \in \mathcal{W}_k$ and zero otherwise. Thirdly, if $[y_{k+1}]_i < 0$ for some i, we can improve the objective function by *deleting* constraint j from $\mathcal{W}_k$, where j is the i-th member of $\mathcal{W}_k$, and thus set $\mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{j\}$. To see this, consider the direction s for which $A_k s = e_i$ and hence $\langle a_j, s \rangle = 1$, and let α be a small positive scalar. Then if follows from the definition of $q(x)$ and (4.12) that

$$\begin{aligned} q(x_{k+1} + \alpha s) - q(x_{k+1}) &= \alpha \langle s, g + Hx_{k+1} \rangle + \tfrac{1}{2}\alpha^2 \langle s, Hs \rangle = -\alpha \langle s, A_k^T y_{k+1} \rangle + \tfrac{1}{2}\alpha^2 \langle s, Hs \rangle \\ &= -\alpha \langle A_k s, y_{k+1} \rangle + \tfrac{1}{2}\alpha^2 \langle s, Hs \rangle = -\alpha [y_{k+1}]_i + \tfrac{1}{2}\alpha^2 \langle s, Hs \rangle < 0 \end{aligned}$$

and

$$\langle a_j, x_{k+1} + \alpha s \rangle = \langle a_j, x_{k+1} \rangle + \alpha \langle a_j, s \rangle = [b]_j + \alpha > [b]_j$$

for all sufficiently small α. Thus the objective can be improved without violating constraint j if we remove the constraint from $\mathcal{W}_k$.

We illustrate the behaviour of the primal active-set method in Figure 4.3. Those with a working knowledge of the Simplex method for linear programming should recognise the basic elements just described, although for this famous algorithm a single iteration combines two of ours: a step followed by the exchange of a constraint encountered with one no longer needed.

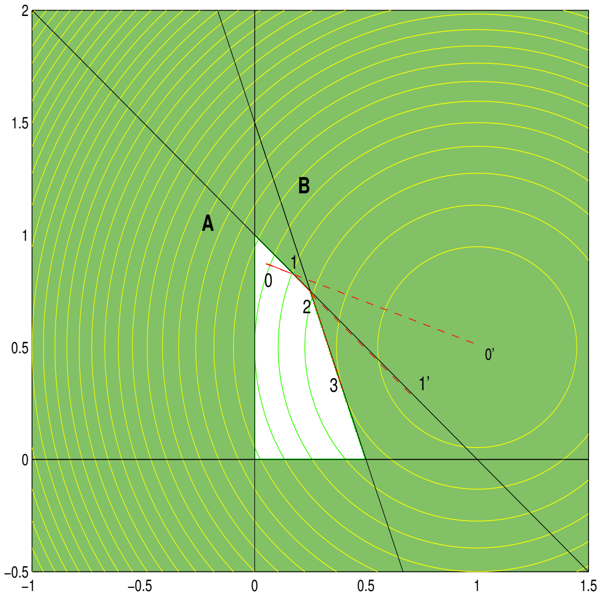

Figure 4.3: Progress of an active-set method on the problem of minimizing $(x_1 - 1)^2 + (x_2 - 0.5)^2$ subject to constraints A: $x_1 + x_2 \leq 1$, B: $3x_1 + x_2 \leq 1.5$ and $(x_1, x_2) \geq 0$. Key: 0: starting point; 0': unconstrained minimizer; 1: encounter constraint A; 1': minimizer on constraint A; 2: encounter constraint B, evaluate Lagrange multipliers, move off constraint A; 3: minimizer on constraint B = required solution.

4.3.2.2 Exploiting the linear algebra—updating factors

Now that we understand the basic mechanics of the primal active-set method, there is one further issue that is vital to its computational efficiency. As we have described it, every iteration requires the solution of an equality-constrained quadratic program (4.11) or equivalently (when H is second-order sufficient over A_k) the linear system (4.13). But the sequence of EQPs/linear systems are closely related in the sense that

$$\begin{aligned} \text{either (i)} \quad & \mathcal{W}_{k+1} = \mathcal{W}_k + \{j\} \quad \text{in which case} \quad A_{k+1} = \begin{pmatrix} A_k \\ a_j^T \end{pmatrix} \\ \text{or (ii)} \quad & \mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{j\} \quad \text{in which case} \quad A_k = \begin{pmatrix} A_{k+1} \\ a_j^T \end{pmatrix}, \end{aligned} \tag{4.14}$$

that is, at the end of each iteration the matrix A_k is changed by adding or removing a single row. Because of this gradual evolution in working sets, our aim is then to update factorizations relating to A_k rather than computing them afresh, since this usually results in a significant saving in algebraic costs.

For the matrices H and $A_k H^{-1} A_k^T$ required by range-space methods this is relatively easy. We assume that we have factorization of H, or some other means of inverting it—clearly H does not change from iteration to iteration. For the other matrix involved, we need Cholesky factors of $L_{k+1} L_{k+1}^T = A_{k+1} H^{-1} A_{k+1}^T$ given those of $L_k L_k^T = A_k H^{-1} A_k^T$. When adding a constraint to the working set, we have

$$A_{k+1}H^{-1}A_{k+1}^T = \begin{pmatrix} A_k H^{-1} A_k^T & A_k H^{-1} a_j \\ a_j^T H^{-1} A_k^T & a_j^T H^{-1} a_j \end{pmatrix},$$

from which it follows immediately that

$$L_{k+1} = \begin{pmatrix} L_k & 0 \\ l^T & \lambda \end{pmatrix}, \text{ where } L_k l = A_k H^{-1} a_j \text{ and } \lambda = \sqrt{a_j^T H^{-1} a_j - l^T l}.$$

Removing a constraint essentially reverses this process.

For null-space methods, the matrices of concern are the null-space basis matrix S_k and the reduced Hessian $S_k^T H S_k$. If we focus on the orthonormal-basis approach, our first need is thus for factors of $A_{k+1} = (L_{k+1} \;\; 0) Q_{k+1}$ given

$$A_k = (L_k \;\; 0) Q_k = (L_k \;\; 0) \begin{pmatrix} Q_{1\,k} \\ Q_{2\,k} \end{pmatrix}$$

for triangular L_k and orthonormal Q_k. To add a constraint (removal is similar), we see that

$$\begin{aligned} A_{k+1} = \begin{pmatrix} A_k \\ a_j^T \end{pmatrix} &= \begin{pmatrix} L_k & 0 \\ a_j^T Q_{1\,k}^T & a_j^T Q_{2\,k}^T \end{pmatrix} Q_k \\ &= \begin{pmatrix} L_k & 0 \\ a_j^T Q_{1\,k}^T & a_j^T Q_{2\,k}^T \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & U_k^T \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & U_k \end{pmatrix} Q_k \\ &= \underbrace{\left[\begin{pmatrix} L_k & 0 \\ a_j^T Q_{1\,k}^T & \sigma e_1^T \end{pmatrix} \right]}_{(L_{k+1} \;\; 0)} \underbrace{\left[\begin{pmatrix} I & 0 \\ 0 & U_k \end{pmatrix} Q_k \right]}_{Q_{k+1}} \end{aligned}$$

where the Householder matrix U_k reduces $Q_{2\,k} a_j$ to σe_1—a *Householder* matrix is an orthonormal matrix of the form $U = I - 2uu^T/\|u\|_2^2$, and may be used to reduce a given vector a to $Ua = \pm\|a\|_2 e_1$ by picking $u = a \pm \|a\|_2 e_1$ where the $\pm$ sign is chosen to guarantee good numerical behaviour. The other matrix we need to factorize is $Q_{2\,k+1} H Q_{2\,k+1}^T$ given known Cholesky factors of $Q_{2\,k} H Q_{2\,k}^T$. Since $Q_{2\,k+1}$ is simply $U_k Q_{2\,k}$ with its first row removed, and as U_k has such special (Householder) structure, it should come as no surprise to the reader that the new required Cholesky factors are cheap to compute—the details are slightly complicated, so we omit them here.

Finally, for the full-space approach, the sole matrix of interest is

$$K_k = \begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix}.$$

To motivate how we might exploit changes in the working set, consider iterations k and $\ell > k$, and suppose that

$$A_k = \begin{pmatrix} A_C \\ A_D \end{pmatrix} \text{ and } A_\ell = \begin{pmatrix} A_C \\ A_A \end{pmatrix},$$

that is, that rows A_D have been deleted and A_A added between the two iterations. In this case, solving

$$\begin{pmatrix} H & A_\ell^T \\ A_\ell & 0 \end{pmatrix} \begin{pmatrix} s_\ell \\ -y_\ell \end{pmatrix} = \begin{pmatrix} -g_\ell \\ 0 \end{pmatrix} \tag{4.15}$$

is the same as solving

$$\left(\begin{array}{ccc|cc} H & A_C^T & A_D^T & A_A^T & 0 \\ A_C & 0 & 0 & 0 & 0 \\ A_D & 0 & 0 & 0 & I \\ \hline A_A & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \end{array}\right) \left(\begin{array}{c} s_\ell \\ -y_C \\ -y_D \\ -y_A \\ u_\ell \end{array}\right) = \left(\begin{array}{c} -g_\ell \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right) \text{ to get } y_\ell = \left(\begin{array}{c} y_C \\ y_A \end{array}\right). \tag{4.16}$$

But notice that the leading sub-matrix in (4.16) is precisely K_k. Thus we can solve (4.16) using factors of K_k and the *Schur complement*

$$S_\ell = -\left(\begin{array}{ccc} A_A & 0 & 0 \\ 0 & 0 & I \end{array}\right) \left(\begin{array}{cc} H & A_k^T \\ A_k & 0 \end{array}\right)^{-1} \left(\begin{array}{cc} A_A^T & 0 \\ 0 & 0 \\ 0 & I \end{array}\right). \tag{4.17}$$

This suggests that we organize the computation into *major* iterations. If a major iteration starts at (ordinary) iteration k, a factorization of K_k will be required. For subsequent iterations ℓ within the current major iteration, (4.15) will be solved using the factors of K_k and the $\ell - k$ by $\ell - k$ Schur complement S_ℓ. Crucially, as $\mathcal{W}_k$ changes gradually to $\mathcal{W}_\ell$, the factorization of S_ℓ should be *updated* not recomputed; the matrix grows by appending one row and column per iteration, so updating is straightforward. Once the dimension of S_ℓ exceeds a given threshold, or it is cheaper to factorize/use K_ℓ than to maintain/use K_k and S_ℓ, this should signal the end of the current major iteration.

4.3.2.3 Other issues

There are three other important issues we have alluded to in our general presentation of the (primal) active-set approach. The first is that these methods need to start from a feasible point. To find an initial feasible point x_0 such that $Ax_0 \geq b$, we could simply use a traditional Simplex *phase-one* procedure as used for linear programming—this and the remaining methods we shall mention also correctly report when the given problem is infeasible. A related alternative is to guess a suitable point, x_{guess}, set $r = \min(b - Ax_{\text{guess}}, 0)$, and then solve the linear program

$$\underset{x\in\mathbb{R}^n,\, \xi\in\mathbb{R}}{\text{minimize}} \quad \xi \ \text{ subject to } \ Ax + \xi r \geq b \ \text{ and } \ \xi \geq 0 \tag{4.18}$$

starting from $(x, \xi) = (x_{\text{guess}}, 1)$, which is feasible for (4.18). As soon as (x_0, ξ_0) is found for which $\xi_0 = 0$, the corresponding x_0 will be feasible for the original problem. However, neither of these approaches pays attention to the true objective function. Two other possibilities which do are *single phase* methods. The first of these simply aims to

$$\underset{x\in\mathbb{R}^n,\, \xi\in\mathbb{R}}{\text{minimize}} \quad q(x) + M\xi \ \text{ subject to } \ Ax + \xi r \geq b \ \text{ and } \ \xi \geq 0$$

for some sufficiently large constant M—naturally this is known the *big-M*, and it is possible to show that it will succeed in most cases. The last of our possibilities is to try to

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad q(x) + \rho \| \max(b - Ax, 0) \|$$

for some sufficiently large ρ. Although this looks at first sight to be a non-differentiable problem, it may actually be reformulated as a QP and solved as such. We illustrate the contours of this function in Figure 4.4.

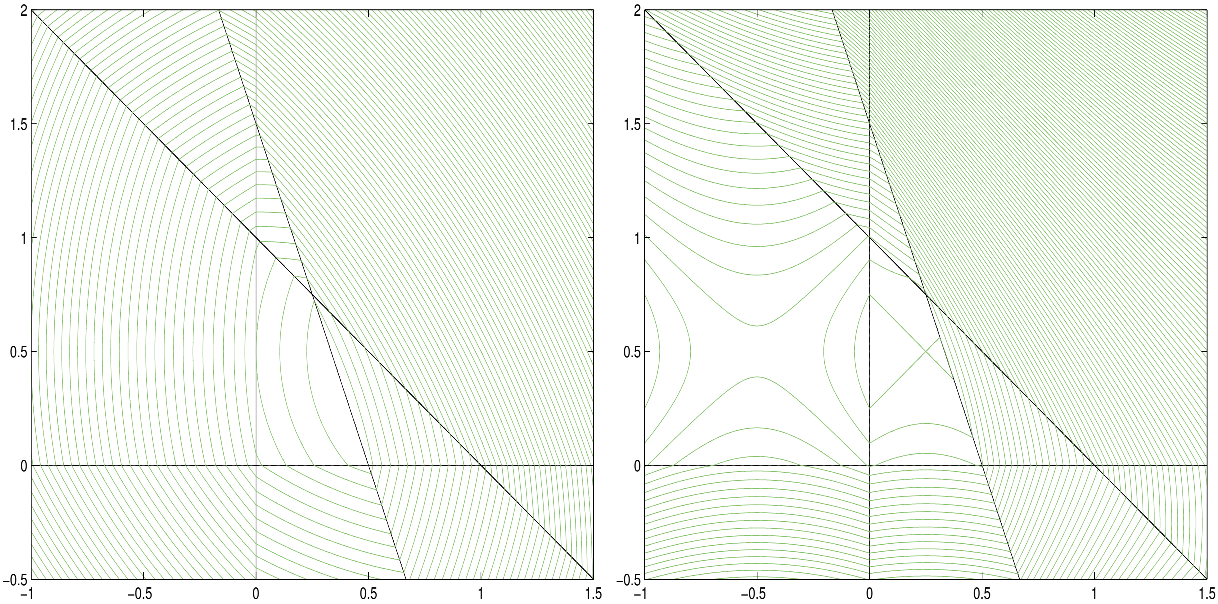

Figure 4.4: Contours of penalty function $q(x) + \rho\|\max(b - Ax, 0)\|$ (left, (with $\rho = 2$) for the convex problem $\min(x_1 - 1)^2 + (x_2 - 0.5)^2$ and (right, with $\rho = 3$) for the non-convex problem $\min -2(x_1 - 0.25)^2 + 2(x_2 - 0.5)^2$, subject to $x_1 + x_2 \leq 1$, $3x_1 + x_2 \leq 1.5$ and $(x_1, x_2) \geq 0$

The second issue we need to consider is convergence of the active-set approach. For a given feasible point, it will only take at most n iterations to find an EQP with a feasible solution, since constraints are only added to the working set until this is so and this requirement is definitely satisfied when the working set contains n constraints. Moreover there are only a finite (but admittedly large) number, 2^m, of possible EQPs with feasible solutions. Finally, so long as a strictly positive step is taken when moving away from the solution to a given EQP, a point with a value objective lower than its optimal will be attained, and thus the EQP will never be revisited. Hence, these methods are finite so long as. $\alpha_k > 0$. If x_k is *degenerate*, that is the active constraints are dependent there, it is possible that $\alpha_k = 0$. If this happens infinitely often, the algorithm might may make no progress, but *cycle*) indefinitely. Various anti-cycling rules—Wolfe's and lexicographic perturbations, Bland's least-index rule and Fletcher's robust method to name a few—have been proposed, and all ensure that the primal active-set method will terminate successfully.

The final issue is how to adapt the methods to cope with non-convexity. Actually, non-convexity causes little extra difficulty so long as suitable factorizations are possible. The main additional idea is inertia control. Briefly, *inertia-controlling* methods tolerate at most one negative eigenvalue in the reduced Hessian $S_k^T H S_k$. The idea is start from working set on which the EQP is strictly convex (for example, a vertex). If a negative eigenvalue subsequent appears as the working sets changes—only one negative eigenvalue can appear per iteration as the working set changes, and this only for iterations for which a constraint is deleted—no further constraint deletions are allowed until convexity is restored. Moreover, for iterations for which there is a negative eigenvalue in the reduced Hessian, it is easy to find a direction of negative curvature in which the EQP decreases. Thus convexity will ultimately be restored, perhaps, but not necessarily, at a (lower) vertex. We note that the latest methods are not inertia controlling, and this allows them to be more flexible with the same convergence guarantees.

Although we have concentrated here on active-set methods, they are no longer necessarily considered to be the best approaches. In particular, when the problem is convex, there are (interior-point) algorithms (see Part 6) that will solve QP in a polynomial number of iterations. To date, there are no known polynomial active-set algorithms. When the problem is non-convex, it is unlikely that there are polynomial algorithms; technically speaking, the problem is NP complete, and even verifying that a proposed solution is locally optimal is NP hard.

4.4 Non-quadratic objectives

We do not plan to say much about the general case for which $f(x)$ is not quadratic. Clearly the main difficulty is that the Hessian now depends on x. The appropriate generic active-set subproblem is to find x_{k+1} to approximately

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) \;\text{ subject to }\; \langle a_i, x\rangle = [b]_i \;\text{ for all }\; i \in \mathcal{W}_k \tag{4.19}$$

for some working set W_k. Some (inner) iteration will now be required to solve (4.19), but crucially each step satisfies $A_k s = 0$ and thus many of the linear algebraic tricks we learned earlier are still valid. It is usual to aim to solve (4.19) inaccurately because to do otherwise for a potentially large number of different active sets would likely be too expensive. But this raises issues of when when to stop each inner iteration and how to compute Lagrange multiplier (estimates) in this case. And stopping early does not prevent the current working set reappearing later on, a phenomenon known colourfully as *zig-zagging*. Zig-zagging is also a possibility for another reason, namely that (4.19) may have many local minimizers.

PART 5

PENALTY AND AUGMENTED LAGRANGIAN METHODS FOR EQUALITY CONSTRAINED OPTIMIZATION

Having given a break-neck description of methods for unconstrained and linearly-constrained minimization, we now turn our attention to the real problems of interest, namely those involving constraints. This part(and Part 7) will focus on problems involving equality constraints, while its successor will be concerned with inequalities. But before we start, we need to discuss the conflicting nature of general constrained optimization problems, and how we might deal with them.

Unconstrained minimization is "simple" because there is but one goal, namely to minimize the objective. This is not so for constrained minimization because there is now a conflict of requirements, the aforementioned objective minimization but at the same time a requirement of feasibility of the solution. While in some instances (such as we have seen for linear equality constraints and, to a certain extent, all inequality constraints) it may be possible to generate feasible iterates, and thus to regain the advantages of having a single goal, this is not usually true for general constrained optimization.

5.1 Merit functions for constrained minimization

Most (but not all, see Part 7.4.3) nonlinearly constrained optimization techniques overcome this dichotomy by introducing a merit function to try to balance the two conflicting requirements of minimization and feasibility. Given parameters p, a composite function $\Phi(x, p)$ is a *merit function* if (some) minimizers of $\Phi(x, p)$ with respect to x approach those of $f(x)$ subject to the constraints as p approaches some set $\mathcal{P}$. Thus a merit function combines both optimality requirements into a single "artificial" objective function. In principal, it then only remains to use the best *unconstrained* minimization methods to solve the constrained problem. If only life were that simple!

In this section, we consider the case of equality constrained minimization, that is finding x_* to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \;\; \text{subject to} \;\; c(x) = 0. \tag{5.1}$$

A suitable merit function in this case is the *quadratic penalty function*

$$\Phi(x, \mu) = f(x) + \frac{1}{2\mu}\|c(x)\|_2^2, \tag{5.2}$$

where μ is a positive scalar *penalty parameter*. It is easy to believe that if μ is small and we try to minimize $\Phi(x, \mu)$ much of the effort will be concentrated on making the second objective term $\frac{1}{2\mu}\|c(x)\|_2^2$ small, that is in forcing $c(x)$ to be small. But as f has a slight presence in the merit function, any remaining energy will be diverted to making $f(x)$ small amongst all of the values for which $c(x)$ is. Formally, as we shall see, it is easy to show that, under modest conditions, some minimizers of $\Phi(x, \mu)$ converge to solutions of (5.1) as μ approaches the set $\{0\}$ from above. We illustrate the appearance of the quadratic penalty function as μ shrinks in Figure 5.1.

Unfortunately, it is possible that $\Phi(x, \mu)$ may have other stationary points that are not solutions of (5.1)—indeed this must be the case if $c(x) = 0$ are inconsistent. Nevertheless, quadratic penalty and related methods are an interesting early development, and we examine them in more detail in this part.

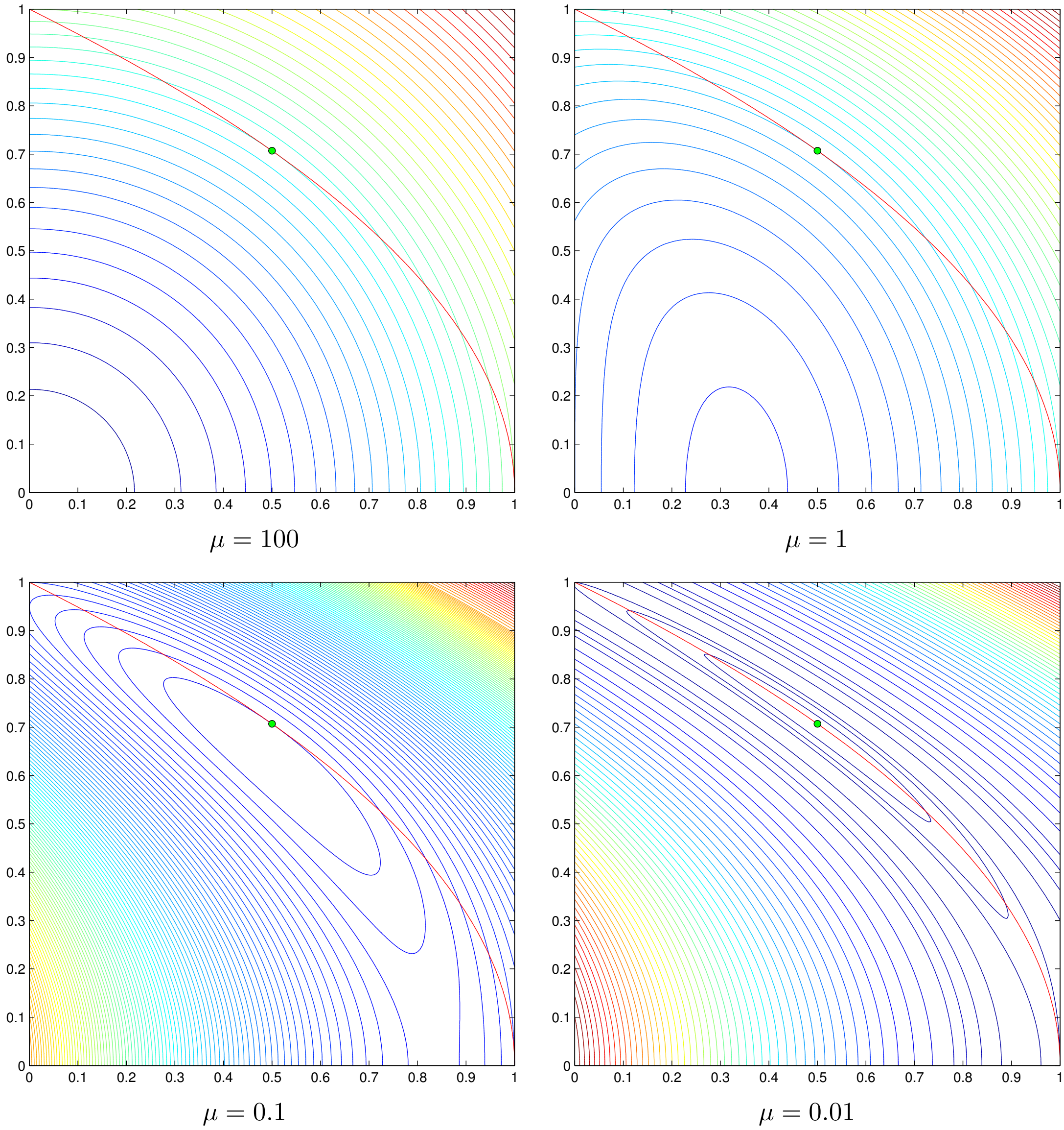

Figure 5.1: The quadratic penalty function for $\min x_1^2 + x_2^2$ subject to $x_1 + x_2^2 = 1$. Notice how the contours become increasingly steep either side of the constraint (red line) as μ shrinks, but rise modestly for points which satisfy it.

5.2 Quadratic penalty methods

Having observed the effect of decreasing the penalty parameter on the minimizers of $\Phi(x, \mu)$, we now make use of this by defining a basic quadratic penalty algorithm.

> Given $\mu_0 > 0$, set $k = 0$
> Until "convergence" iterate:
> Starting from x_k^{S}, use an unconstrained minimization algorithm to find an "approximate" minimizer x_k of $\Phi(x, \mu_k)$
> Compute $\mu_{k+1} > 0$ smaller than μ_k such that $\lim_{k\to\infty} \mu_{k+1} = 0$ and increase k by 1

In practice it is common to choose $\mu_{k+1} = 0.1\mu_k$ or even $\mu_{k+1} = \mu_k^2$, while the obvious choice for a subsequent starting point is $x_{k+1}^{\text{S}} = x_k$.

Fortunately, as we have hinted, basic convergence of this algorithm is easily established. To do so, we define what are known as *first-order Lagrange multiplier estimates* for the quadratic penalty function,

$$y(x, \mu) \stackrel{\text{def}}{=} -\frac{c(x)}{\mu}.$$

Then we have following global convergence result.

Theorem 5.1. Suppose that f, $c \in \mathcal{C}^2$, that

$$\|\nabla_x \Phi(x_k, \mu_k)\|_2 \leq \epsilon_k, \tag{5.3}$$

where ϵ_k and μ_k converge to zero as k increases, that $y_k = y(x_k, \mu_k)$ and that x_k converges to x_* for which $A(x_*)$ is full rank. Then x_* satisfies the first-order necessary optimality conditions for the equality-constrained problem (5.1) and $\{y_k\}$ converge to the associated Lagrange multipliers y_*.

Thus we see that, so long as the limiting constraint Jacobian is well behaved, it suffices to (approximately) minimize $\Phi(x, \mu)$—if ever the full-rank assumption is violated at x_*, it is easy to show that x_* (locally) minimizes the infeasibility $\|c(x)\|_2$, so a locally closest feasible point is found. Since $\Phi(x, \mu)$ is a smooth function, we can immediately appeal to the methods we derived in Parts 2 and 3. But as the illustrations in Figure 5.1 might have already suggested, some care is needed.

One thing is immediately apparent from the figure; as μ becomes small, the contours elongate in directions tangential to the (gradients of the) constraints or, alternatively, the function becomes very steep normal to the constraints. Thus increasingly rapid change is more likely for steps in normal rather than tangential directions. This has particularly serious implications for trust-region methods, where the shape of the trust-region needs to recognise that any reasonable model of $\Phi(x, \mu)$ will reflect this behaviour. We shall return to this shortly.

It is instructive to consider the derivatives of $\Phi(x, \mu)$. We have

$$\begin{aligned} \nabla_x \Phi(x, \mu) &= g(x, y(x, \mu)) \text{ and} \\ \nabla_{xx} \Phi(x, \mu) &= H(x, y(x, \mu)) + \frac{1}{\mu} A^T(x) A(x), \end{aligned} \tag{5.4}$$

where $g(x, y)$ and $H(x, y)$ are the gradient (1.2) and Hessian (1.3) of the Lagrangian function (1.1). As μ shrinks, the first term in (5.4) is usually well behaved, but the second term clearly diverges and ultimately dominates the Hessian, at least in the subspace spanned by the columns of $A^T(x)$. More formally we can show the following.

Theorem 5.2. Suppose that the assumptions of Theorem 5.1 are satisfied. Then the Hessian matrix of the quadratic penalty function, $\nabla_{xx} \Phi(x_k, \mu_k)$, for the equality-constrained problem (5.1) involving m constraints has m eigenvalues

$$\lambda_i(A(x_*) A^T(x_*)/\mu_k + O(1) \text{ for } i = 1, \ldots, m$$

and the remaining $n - m$ eigenvalues

$$\lambda_i(S^T H(x_*, y_*) S) + O(\mu_k) \text{ for } i = 1, \ldots, n - m$$

as $k \to \infty$, where $\lambda_i(.)$ denotes the i-th eigenvalue of its matrix argument and S is an orthogonal basis for the null-space of $A(x_*)$.

Thus as our pictures suggested, the Hessian matrix becomes increasingly and inevitably ill conditioned as the quadratic penalty algorithm converges, in all but the exceptional cases where $m = 0$ or n—by *ill conditioning*, here we mean simply that the ratio of smallest to largest eigenvalue of $\nabla_{xx} \Phi(x_k, \mu_k) = O(1/\mu_k)$ is large. Since ultimately we would naturally hope to use some variant of Newton's method

$$\nabla_{xx} \Phi(x, \mu) s = -\nabla_x \Phi(x, \mu) \tag{5.5}$$

when finding a correction s to a current estimate x of x_*, we might be concerned that the ill conditioning might thwart our efforts to solve (5.5) accurately. After all, aren't we taught in numerical analysis classes that even stable factorization methods produce inaccurate results for ill-conditioned linear systems? Only if we aren't paying enough attention! In particular, such results are typically true for general right-hand side of the linear system under consideration, but there are specific right-hand sides that do not suffer. And (5.5) is just such an example.

One way of seeing this is to write (5.5) as

$$\left(H(x, y(x, \mu)) + \frac{1}{\mu} A^T(x) A(x) \right) s = -\left(g(x) + \frac{1}{\mu} A^T(x) c(x) \right) \tag{5.6}$$

and to define auxiliary variables w for which

$$w = \frac{1}{\mu}\left(A(x)s + c(x)\right).$$

This then implies

$$\begin{pmatrix} H(x,y(x,\mu)) & A^T(x) \\ A(x) & -\mu I \end{pmatrix}\begin{pmatrix} s \\ w \end{pmatrix} = -\begin{pmatrix} g(x) \\ c(x) \end{pmatrix}. \tag{5.7}$$

But this equation is essentially independent of μ for small μ and thus cannot suffer for ill-conditioning inherited from μ. Thus it is possible to find s accurately by solving (5.6)—a more sophisticated analysis is needed to show that the same is true for the original formulation (5.5).

Nonetheless, the ill-conditioning of $\nabla_{xx}\Phi(x,\mu)$ is an inevitable fact, and globally $\Phi(x,\mu)$ will be hard to minimize for small μ if started from an arbitrary point. It is the fact that the iterate x_k described in our algorithmic sketch is already close to x_{k+1} which makes a sequential minimization a far more appealing process than a one-off minimization of $\Phi(x,\mu)$ with a small μ from an arbitrary starting point x. Of course, what "small" means is problem dependent, and for early minimizations in our algorithmic framework we still need to be concerned that our unconstrained minimization algorithm can cope with poorly conditioned Hessians. To combat this, trust-region methods should aim to encourage equal model decreases in all allowed directions, and this is best accomplished by using a weighted norm $\|v\|_B = \sqrt{\langle v, Bv\rangle}$ for some approximation B to $\nabla_{xx}\Phi(x,\mu)$ which reflects the ill-conditioned terms. This has the effect of prohibiting relatively large steps in directions normal to the constraint gradients.

5.3 Perturbed optimality conditions

It is worth considering a seemingly alternative method for solving our problem at this stage. We know from Theorem (1.7) that the first order optimality conditions for (5.1) are that there are Lagrange multipliers y for which

$$\begin{array}{rl} g(x) - A^T(x)y = 0 & \text{dual feasibility} \\ c(x) = 0 & \text{primal feasibility} \end{array} \tag{5.8}$$

Now consider the "perturbed" problem

$$\begin{array}{rl} g(x) - A^T(x)y = 0 & \text{dual feasibility} \\ c(x) + \mu y = 0 & \textit{perturbed}\text{ primal feasibility} \end{array} \tag{5.9}$$

where $\mu > 0$. Although it is possible simply to apply a (safeguarded) Newton method to (5.8)—and indeed this is the theme of Part 7—here we consider doing the same to (5.9).

Given an approximation (x,y) to a root of (5.9), the Newton correction (s,v) satisfies

$$\begin{pmatrix} H(x,y) & -A^T(x) \\ A(x) & \mu I \end{pmatrix}\begin{pmatrix} s \\ v \end{pmatrix} = -\begin{pmatrix} g(x) - A^T(x)y \\ c(x) + \mu y \end{pmatrix}$$

If we eliminate w from these equations, we see that

$$\left(H(x,y) + \frac{1}{\mu}A^T(x)A(x)\right)s = -\left(g(x) + \frac{1}{\mu}A^T(x)c(x)\right). \tag{5.10}$$

But a quick comparison of (5.6) with (5.10) reveals that the only difference is that the former is constrained to pick $y(x, mu)$ as Lagrange multiplier estimates in the Hessian of Lagrangian, while the latter may use arbitrary y. Thus there is a very strong connection between following paths of the perturbed optimality conditions and the quadratic penalty function method. This observation may be made for other merit-function based methods, and will be crucial in Part 6.

5.4 Augmented Lagrangian methods

A second, related, merit function for the equality-constrained problem (5.1) is the *augmented Lagrangian function*

$$\Phi(x, u, \mu) = f(x) - u^T c(x) + \frac{1}{2\mu}\|c(x)\|_2^2$$

where both u and μ are auxiliary parameters. This function can be viewed either as a convexification of the Lagrangian function —hence the name—or as a shifted quadratic penalty function. Although there are strong arguments from duality theory to support the former interpretation, it is the latter that we prefer here. Our aim, as before, is to adjust μ *and* u to encourage convergence.

In the case of the augmented Lagrangian function,

$$y(x, u, \mu) \stackrel{\text{def}}{=} u - \frac{c(x)}{\mu}.$$

give suitable first-order Lagrange multiplier estimates. For then

$$\begin{aligned} \nabla_x \Phi(x, u, \mu) &= g(x, y(x, u, \mu)) \text{ and} \\ \nabla_{xx} \Phi(x, u, \mu) &= H(x, y(x, u, \mu)) + \frac{1}{\mu} A^T(x) A(x) \end{aligned} \tag{5.11}$$

Clearly we recover the quadratic penalty function when $u = 0$, but, as the following result shows, there may be more suitable choices.

Theorem 5.3. Suppose that $f,\, c \in \mathcal{C}^2$, that

$$\|\nabla_x \Phi(x_k, u_k, \mu_k)\|_2 \le \epsilon_k,$$

where ϵ_k converges to zero as k increases, that $y_k = y(x_k, u_k, \mu_k)$ for given $\{u_k\}$ and $\{\mu_k\}$, and that x_k converges to x_* for which $A(x_*)$ is full rank. Then $\{y_k\}$ converge to some y_* for which $g(x_*) = A^T(x_*) y_*$.

If additionally either μ_k converges to zero for bounded u_k or u_k converges to y_* for bounded μ_k, x_* and y_* satisfy the first-order necessary optimality conditions for the equality-constrained problem (5.1).

Thus there are now two ways to encourage convergence to a solution to our problem, one mimicing the quadratic penalty function but, more significantly, another by encouraging the parameters u_k to approach the desired Lagrange multipliers *without* forcing μ_k to zero. This latter approach is highly desirable, as then the Hessian (5.11) will not inevitably become ill-conditioned. We illustrate this new approach in Figure 5.2.

Since y_* is not known in advance, It is not immediately obvious how we might choose suitable u_k, but a moment's reflection indicates that the algorithm itself provides suitable estimates. We know that convergence is guaranteed if u_k is fixed while μ_k decreases to zero, and in this case y_k will converge to y_* and $c(x_k)$ to zero. Thus if both $c(x_k)$ and $\nabla_x \Phi(x_k, u_k, \mu_k)$ are small, there is good reason to believe that (x_k, y_k) are close to (x_*, y_*), and thus reasonable to assign $u_{k+1} = y_k$.

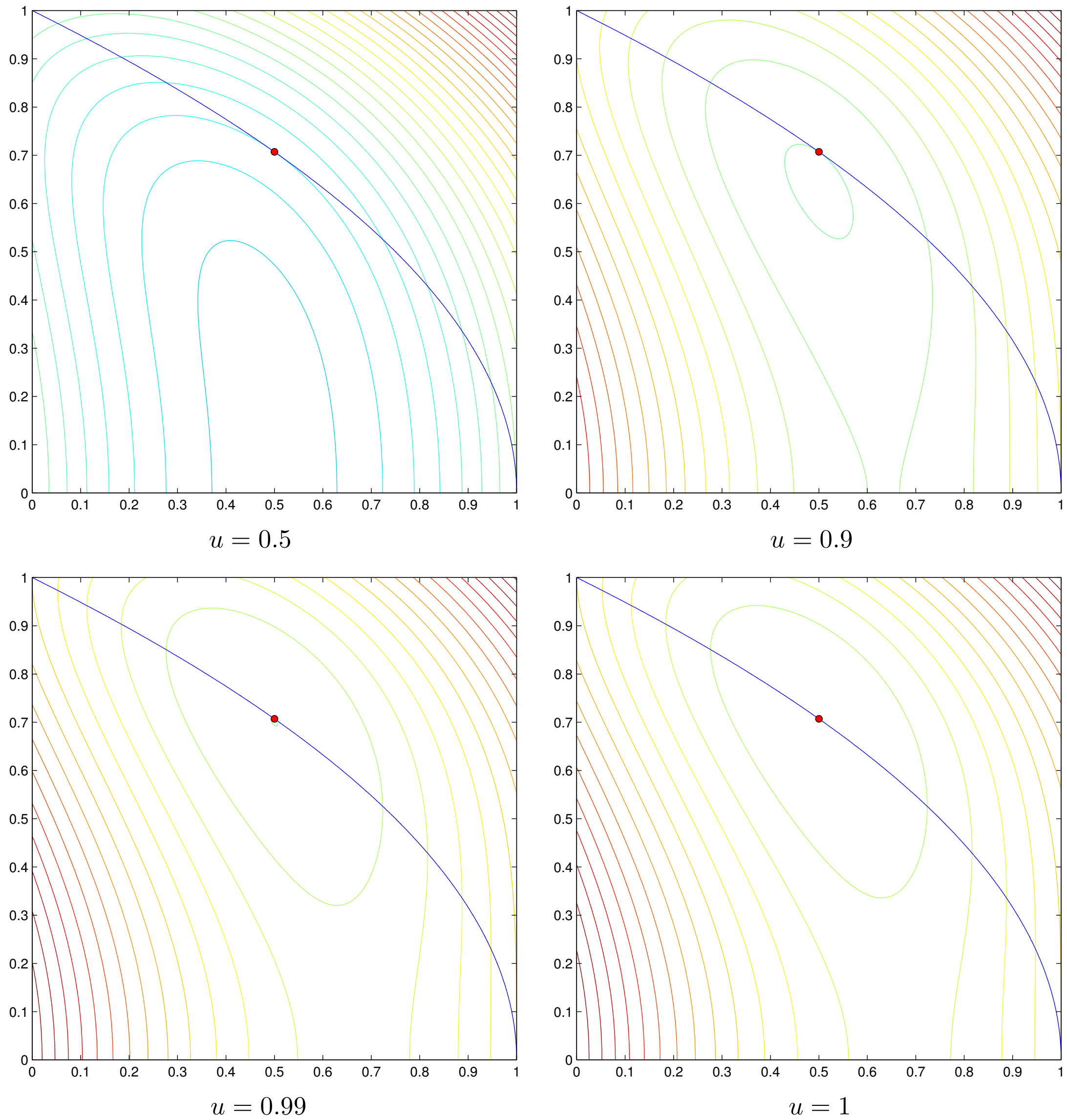

Figure 5.2: The augmented Lagrangian function for $\min x_1^2 + x_2^2$ subject to $x_1 + x_2^2 = 1$ with μ fixed at 1 as u varies. Now notice that the contours are modest, and how the minimizer of the augmented Lagrangian function moves towards $x_* = (\frac{1}{2}, \frac{\sqrt{3}}{2})$ as u approaches $y_* = 1$.

Formally, we might check if $\|c(x_k)\| \leq \eta_k$ where $\{\eta_k\}$ is a positive sequence with limit zero, and if so set $u_{k+1} = y_k$ and $\mu_{k+1} = \mu_k$. If this test fails, then we may not yet be close enough to believe that y_k is a good estimate of y_*, and our only recourse is to try to move closer by reducing μ without changing u. We summarize this in the following augmented Lagrangian algorithm.

> Given $\mu_0 > 0$ and u_0, set $k = 0$
> Until "convergence" iterate:
> Starting from x_k^{s}, use an unconstrained minimization algorithm to find an "approximate" minimizer x_k of $\Phi(x, u_k, \mu_k)$ for which $\|\nabla_x \Phi(x_k, u_k, \mu_k)\| \leq \epsilon_k$
> If $\|c(x_k)\| \leq \eta_k$, set $u_{k+1} = y_k$ and $\mu_{k+1} = \mu_k$
> Otherwise set $u_{k+1} = u_k$ and $\mu_{k+1} \leq \tau \mu_k$
> Set suitable ϵ_{k+1} and η_{k+1} and increase k by 1

In practice suitable values for these tolerances might be

$$\epsilon_k = \mu_k^j \quad \text{and} \quad \eta_k = \mu_k^{0.1+0.9j},$$

where j iterations have passed since μ was last decreased, while $\tau = \min(0.1, \sqrt{\mu_k})$ is suitable for encouraging fast linear convergence. Significantly, under rules like these, it can be shown that the penalty parameter will ultimately remain unchanged if x_k converges to a single limit point. One other important issue is that, even under second-order sufficiency conditions, the Hessian (5.11) of $\Phi(x, u, \mu)$ is only guaranteed to be positive definite for sufficiently small μ, but that our augmented Lagrangian algorithm does not explicitly guarantee this. Thus in practice extra precautions may be required to reduce μ more than simply as described in our algorithm.

PART 6

INTERIOR-POINT METHODS FOR INEQUALITY CONSTRAINED OPTIMIZATION

6.1 The logarithmic barrier function for inequality constraints

For the inequality constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \;\; \text{subject to} \;\; c(x) \geq 0 \tag{6.1}$$

the best known merit function is the *logarithmic barrier function*

$$\Phi(x, \mu) = f(x) - \mu \sum_{i=1}^{m} \log c_i(x),$$

where μ is again a positive scalar *barrier parameter*. Each logarithmic term $-\log c_i(x)$ becomes infinite as x approaches the boundary of the i-th inequality from the feasible side, and is undefined (effectively infinite) beyond there. The size of the logarithmic term is mitigated when μ is small, and it is then possible to get close to the boundary of the feasible region before its effect is felt, any minimization effort being directed towards reducing the objective. Once again, it is easy to show that, under modest conditions, some minimizers of $\Phi(x, \mu)$ converge to solutions of (6.1) as μ approaches the set $\{0\}$ from above. And once again a possible defect is that $\Phi(x, \mu)$ may have other, useless stationary points. The contours of a typical example are shown in Figure 6.1.

6.2 A basic barrier-function algorithm

The logarithmic barrier function is different in one vital aspect from the quadratic penalty function in that it requires that there is a *strictly* interior point. If we apply the obvious sequential minimization algorithm to $\Phi(x, \mu)$, a strictly interior starting point is required, and all subsequent iterates will be strictly interior. The obvious "interior-point" algorithm is as follows.

> Given $\mu_0 > 0$, set $k = 0$.
> Until "convergence", iterate:
>   Find x^{S}_k for which $c(x^{\mathrm{S}}_k) > 0$.
>   Starting from x^{S}_k, use an unconstrained minimization algorithm to find an "approximate" minimizer x_k of $\Phi(x, \mu_k)$.
>   Compute $\mu_{k+1} > 0$ smaller than μ_k such that $\lim_{k \to \infty} \mu_{k+1} = 0$. and increase k by 1.

In practice it is common to choose $\mu_{k+1} = 0.1\mu_k$ or even $\mu_{k+1} = \mu_k^2$, while perhaps the obvious choice for a subsequent starting point is $x^{\mathrm{S}}_{k+1} = x_k$.

Fortunately, as we have hinted, basic convergence for the algorithm is easily established. Recall that the active set $\mathcal{A}(x)$ at a point x is $\mathcal{A}(x) = \{i \mid c_i(x) = 0\}$. Just as for the quadratic penalty function, we define first-order Lagrange multiplier estimates for the logarithmic barrier function,

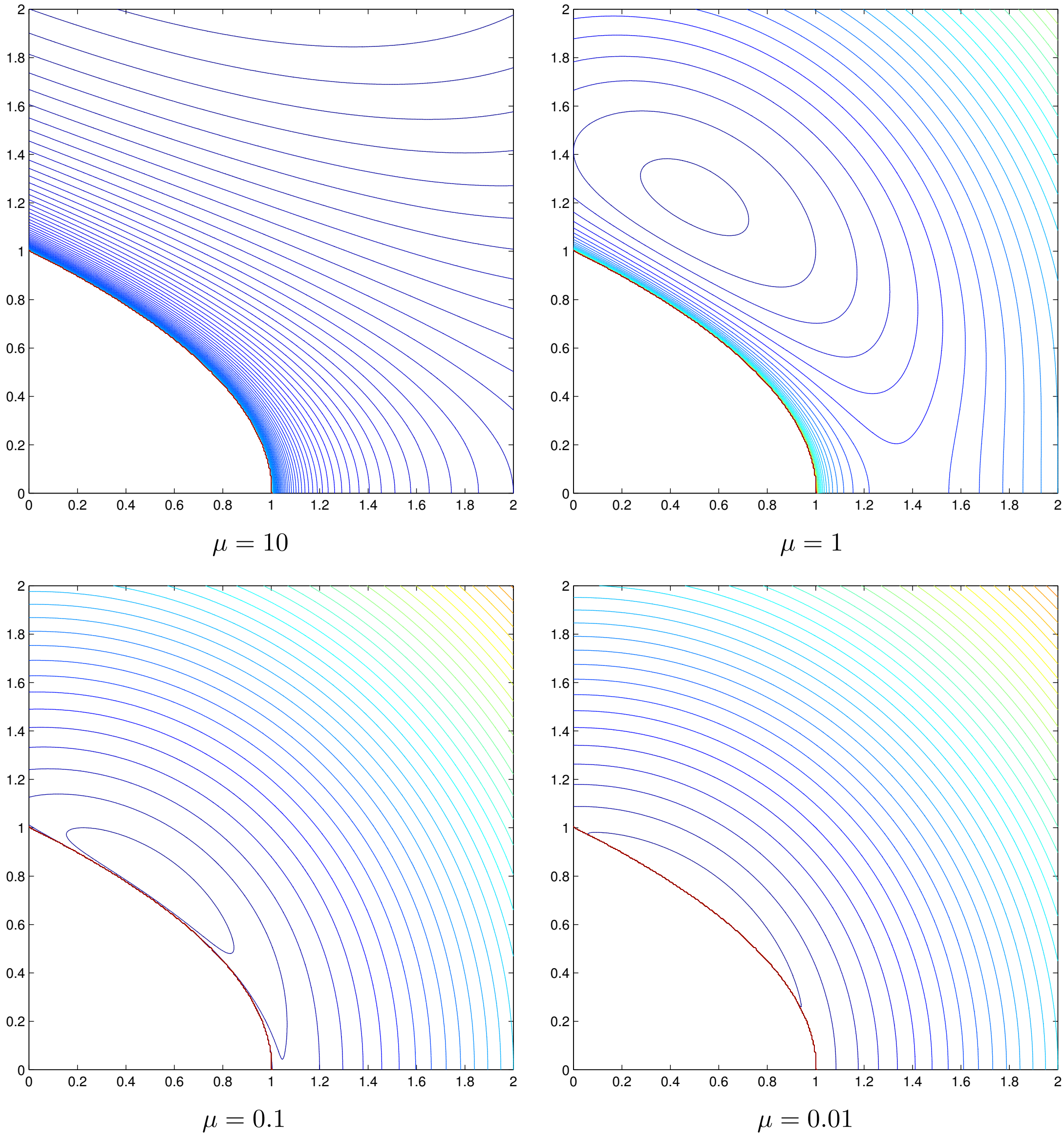

Figure 6.1: The logarithmic barrier function for $\min x_1^2 + x_2^2$ subject to $x_1 + x_2^2 \geq 1$. The contours for $\mu = 0.01$ are visually indistinguishable from $f(x)$ for feasible points.

this time so that

$$y_i(x, \mu) \stackrel{\text{def}}{=} \frac{\mu}{c_i(x)}.$$

Then we have the following.

Theorem 6.1. Suppose that f, $c \in \mathcal{C}^2$, that

$$\|\nabla_x \Phi(x_k, \mu_k)\|_2 \leq \epsilon_k$$

where ϵ_k converges to zero as k increases, that $y_k = y(x_k, \mu_k)$, and that x_k converges to x_* for which $\{a_i(x_*)\}_{i \in \mathcal{A}(x_*)}$ are linearly independent. Then x_* satisfies the first-order necessary optimality conditions for the inequality-constrained problem (6.1) and $\{y_k\}$ converge to the associated Lagrange multipliers y_*.

Notice here how the algorithm delivers something unexpected, namely estimates of the Lagrange multipliers. Also see the role played by the linearly independence of the active constraint gradients, regrettably quite a strong constraint qualification.

6.3 Potential difficulties

As we now know that it suffices to (approximately) minimize $\Phi(x, \mu)$, how should we proceed? As $\Phi(x, \mu)$ is a smooth function, we can immediately appeal to the methods we discussed in Parts 2 and 3. But we need to be careful. Very, very careful.

We could use a linesearch method. Of note here is the fact that the barrier function has logarithmic singularities, indeed is undefined for infeasible points. Thus it makes sense to design a specialized linesearch to cope with the singularity of the log. Alternatively, we could use a trust-region method. Here we need to be able to instantly reject candidate steps for which $c(x_k + s_k) \not> 0$. More importantly, while all (consistent) trust-region norms are equivalent, (ideally) we should "shape" the trust region for any barrier-function model to cope with the contours of the singularity. This implies that the trust-region shape may vary considerably from iteration to iteration, with its shape reflecting the eigenvalues arising from the singularity.

6.3.1 Potential difficulty I: ill-conditioning of the barrier Hessian

At the heart of both linesearch and trust-region methods is, of course, the Newton (second-order) model and related Newton direction. The computation of a Newton model/direction for the logarithmic barrier function is vital, and the resulting equations have a lot of (exploitable) structure. The gradient of the barrier function is

$$\nabla_x \Phi(x, \mu) = g(x) - \mu \sum_i a_i(x)/c_i(x) = g(x) - A^T(x) y(x, \mu) = g(x, y(x, \mu)),$$

where $g(x, y)$ is the gradient (1.2) of the Lagrangian function for (6.1). Likewise, the Hessian is

$$\nabla_{xx} \Phi(x, \mu) = H(x, y(x, \mu)) + \mu A^T(x) C^{-2}(x) A(x),$$

where $H(x, y)$ is the Hessian (1.3) of the Lagrangian and $C(x) = \mathrm{diag}(c_1(x), \ldots, c_m(x))$, the diagonal matrix whose entries are the $c_i(x)$. Thus the Newton correction s^{P} from x for the barrier function satisfies

$$(H(x, y(x, \mu)) + \mu A^T(x) C^{-2}(x) A(x)) s^{\text{P}} = -g(x, y(x, \mu)). \tag{6.2}$$

Since $y(x, \mu) = \mu C^{-1}(x)e$, (6.2) is sometimes written as

$$\left(H(x, y(x, \mu)) + A^T(x) C^{-1}(x) Y(x, \mu) A(x)\right) s^{\text{P}} = -g(x, y(x, \mu)), \tag{6.3}$$

or

$$\left(H(x, y(x, \mu)) + A^T(x) Y^2(x, \mu) A(x)/\mu\right) s^{\text{P}} = -g(x, y(x, \mu)), \tag{6.4}$$

where $Y(x, \mu) = \text{diag}(y_1(x, \mu), \ldots, y_m(x, \mu))$.

This is where we need to be careful. For we have the following estimates of the eigenvalues of the barrier function as we approach a solution.

Theorem 6.2. Suppose that the assumptions of Theorem 6.1 are satisfied, that $A_{\mathcal{A}}$ is the matrix whose rows are $\{a_i^T(x_*)\}_{i \in \mathcal{A}(x_*)}$, that $m_a = |\mathcal{A}(x_*)|$, and that x_* is *non-degenerate*, that is $(y_*)_i > 0$ for all $i \in \mathcal{A}(x_*)$. Then the Hessian matrix of the barrier function, $\nabla_{xx}\Phi(x_k, \mu_k)$, has m_a eigenvalues

$$\lambda_i(Y_{\mathcal{A}} A_{\mathcal{A}} A_{\mathcal{A}}^T Y_{\mathcal{A}})/\mu_k + O(1) \;\; \text{for } i = 1, \ldots, m_a$$

and the remaining $n - m_a$ eigenvalues

$$\lambda_i(N_{\mathcal{A}}^T H(x_*, y_*) N_{\mathcal{A}}) + O(\mu_k) \;\; \text{for } i = 1, \ldots, n - m_a$$

as $k \to \infty$, where $\lambda_i(.)$ denotes the i-th eigenvalue of its matrix argument, $Y_{\mathcal{A}}$ is the diagonal matrix of active Lagrange multipliers at x_* and $N_{\mathcal{A}} =$ is an orthogonal basis for the null-space of $A_{\mathcal{A}}$.

This demonstrates that the condition number of $\nabla_{xx}\Phi(x_k, \mu_k)$ is $O(1/\mu_k)$ as μ_k shrinks to zero, and suggests that it may not be straightforward to find the minimizer numerically. Look at how the contours around x_* in Figure 6.1 bunch together as μ approaches zero.

6.3.2 Potential difficulty II: poor starting points

As if this potential defect isn't serious enough, there is a second significant difficulty with the naive method we described earlier. This is that $x_{k+1}^{\text{S}} = x_k$ appears to be a very poor starting point for a Newton step just after the (small) barrier parameter is reduced. To see this suppose, as will be the case at the end of the minimization for the k-th barrier subproblem, that

$$0 \approx \nabla_x \Phi(x_k, \mu_k) = g(x_k) - \mu_k A^T(x_k) C^{-1}(x_k) e \approx g(x_k) - \mu_k A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-1}(x_k) e,$$

the approximation being true because the neglected terms involve $y(x_k, \mu_k) = \mu_k / c_i(x_k)$ which converge to zero for inactive constraints. Then in the non-degenerate case, again roughly speaking, the Newton correction s^{P} for the new barrier parameter satisfies

$$\mu_{k+1} A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-2}(x_k) A_{\mathcal{A}}(x_k) s^{\text{P}} \approx (\mu_{k+1} - \mu_k) A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-1}(x_k) e \tag{6.5}$$

since

$$\nabla_x \Phi(x_k, \mu_{k+1}) \approx g(x_k) - \mu_{k+1} A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-1}(x_k) e \approx (\mu_{k+1} - \mu_k) A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-1}(x_k) e$$

and the $\mu_{k+1} A_{\mathcal{A}}^T(x_k) C_{\mathcal{A}}^{-2}(x_k) A_{\mathcal{A}}(x_k)$ term dominates $\nabla_{xx}\Phi(x_k, \mu_{k+1})$. If $A_{\mathcal{A}}(x_k)$ is full rank, then multiplying the approximation (6.5) from the left first by the generalized inverse, $(A_{\mathcal{A}} A_{\mathcal{A}}^T)^{-1} A_{\mathcal{A}}$ of $A_{\mathcal{A}}$ and then by $C_{\mathcal{A}}^2$ implies that

$$A_{\mathcal{A}}(x_k) s^{\mathrm{P}} \approx \left(1 - \frac{\mu_k}{\mu_{k+1}}\right) c_{\mathcal{A}}(x_k)$$

from which a Taylor expansion of $c_{\mathcal{A}}(x_k + s^{\mathrm{P}})$ reveals that

$$c_{\mathcal{A}}(x_k + s^{\mathrm{P}}) \approx c_{\mathcal{A}}(x_k) + A_{\mathcal{A}}(x_k) s^{\mathrm{P}} \approx \left(2 - \frac{\mu_k}{\mu_{k+1}}\right) c_{\mathcal{A}}(x_k) < 0$$

whenever $\mu_{k+1} < \frac{1}{2}\mu_k$. Hence a Newton step will asymptotically be infeasible for anything but the most modest decrease in μ, and thus the method is unlikely to converge fast.

We will return to both of these issues shortly, but first we need to examine barrier methods in a seemingly different light.

6.4 A different perspective: perturbed optimality conditions

We now consider what, superficially, appears to be a completely different approach to inequality-constrained optimization. Recall from Theorem (1.9) that the first order optimality conditions for (6.1) are that there are Lagrange multipliers (or, as they are sometimes called, dual variables) y for which

$$\begin{aligned} g(x) - A^T(x) y &= 0 && \text{(dual feasibility)} \\ C(x) y &= 0 && \text{(complementary slackness)} \\ c(x) \geq 0 \;\text{ and }\; y &\geq 0. \end{aligned}$$

Now consider the "perturbed" problem

$$\begin{aligned} g(x) - A^T(x) y &= 0 && \text{(dual feasibility)} \\ C(x) y &= \mu e && \text{(perturbed complementary slackness)} \\ c(x) > 0 \;\text{ and }\; y &> 0, \end{aligned}$$

where $\mu > 0$.

Primal-dual path-following methods aim to track solutions to the system

$$g(x) - A^T(x) y = 0 \;\text{ and }\; C(x) y - \mu e = 0 \tag{6.6}$$

as μ shrinks to zero, while maintaining $c(x) > 0$ and $y > 0$. This approach has been amazingly successful when applied to linear programming problems, and has been extended to many other classes of convex optimization problems. Since (6.6) is simply a nonlinear system, an obvious (locally convergent) way to solve the system is, as always, to use Newton's method. It is easy to show that the Newton correction (s^{PD}, w) to (x, y) satisfies

$$\begin{pmatrix} H(x,y) & -A^T(x) \\ Y A(x) & C(x) \end{pmatrix} \begin{pmatrix} s^{\mathrm{PD}} \\ w \end{pmatrix} = - \begin{pmatrix} g(x) - A^T(x) y \\ C(x) y - \mu e \end{pmatrix}. \tag{6.7}$$

Using the second equation to eliminate w gives that

$$\left(H(x,y) + A^T(x)C^{-1}(x)YA(x)\right) s^{\text{PD}} = -\left(g(x) - \mu A^T(x)C^{-1}(x)e\right) = g(x, y(x,\mu)), \tag{6.8}$$

where, as before, $y(x,\mu) = \mu C^{-1}(x)e$. But now compare this with the Newton barrier system (6.3). Amazingly, the only difference is that the (left-hand-side) coefficient matrix in (6.3) mentions the specific $y(x,\mu)$ while that for (6.8) uses a generic y. And it is this difference that turns out to be crucial. The freedom to choose y in $H(x,y) + A^T(x)C^{-1}(x)YA(x)$ for the primal-dual approach proves to be vital. Making the primal choice $y(x,\mu) = \mu C^{-1}(x)e$ can be poor, while using a more flexible approach in which y is chosen by other means, such as through the primal-dual correction $y + w$ is often highly successful.

We now return to the potential difficulties with the primal approach we identified in Parts 6.3.1 and 6.3.2.

6.4.1 Potential difficulty II ... revisited

We first show that, despite our reservations in Part 6.3.2, the value $x^{\text{S}}_{k+1} = x_k$ can be a good starting point. The problem with the primal correction s^{P} is that the primal method has to choose $y = y(x^{\text{S}}_{k+1}, \mu_{k+1}) = \mu_{k+1}C^{-1}(x_k)e$, and this is a factor μ_{k+1}/μ_k too small to be a good Lagrange multiplier estimate—recall that Theorem 6.1 shows that $\mu_k C^{-1}(x_k)e$ converges to y_*.

But now suppose instead that we use the primal-dual correction s^{PD} and choose the "proper" $y = y(x_k, \mu_k) = \mu_k C^{-1}(x_k)e$ rather than $y(x^{\text{S}}_{k+1}, \mu_{k+1})$—we know that this is a good choice insofar as this Newton step should decrease the dual infeasibility and complementary slackness since $(x_k, \mu_k C^{-1}(x_k)e)$ are already good estimates. In this case, arguing as before, in the non-degenerate case, the correction s^{PD} satisfies

$$\mu_k A^T_{\mathcal{A}}(x_k)C^{-2}_{\mathcal{A}}(x_k)A_{\mathcal{A}}(x_k)s^{\text{PD}} \approx (\mu_{k+1} - \mu_k)A^T_{\mathcal{A}}(x_k)C^{-1}_{\mathcal{A}}(x_k)e,$$

and thus if $A_{\mathcal{A}}(x_k)$ is full rank,

$$A_{\mathcal{A}}(x_k)s^{\text{PD}} \approx \left(\frac{\mu_{k+1}}{\mu_k} - 1\right) c_{\mathcal{A}}(x_k).$$

Then using a Taylor expansion of $c_{\mathcal{A}}(x_k + s^{\text{PD}})$ reveals that

$$c_{\mathcal{A}}(x_k + s^{\text{PD}}) \approx c_{\mathcal{A}}(x_k) + A_{\mathcal{A}}(x_k)s^{\text{PD}} \approx \frac{\mu_{k+1}}{\mu_k} c_{\mathcal{A}}(x_k) > 0,$$

and thus $x_k + s^{\text{PD}}$ is feasible—the result is easy to show for inactive constraints. Hence, simply by using a different model Hessian we can compute a useful Newton correction from $x^{\text{S}}_{k+1} = x_k$ that both improves the violation of the optimality conditions (and ultimately leads to fast convergence) *and* stays feasible.

6.4.2 Primal-dual barrier methods

In order to globalize the primal-dual iteration, we simply need to build an appropriate model of the logarithmic barrier function within either a linesearch or trust-region framework for minimizing $\Phi(x, \mu_k)$. As we have already pointed out the disadvantages of only allowing the (primal) Hessian approximation $\nabla_{xx}\Phi(x_k,\mu_k)$, we instead prefer the more flexible search-direction model problem to (approximately)

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \quad \langle s, g(x,y(x,\mu))\rangle + \tfrac{1}{2}\left\langle s, \left(H(x,y)+A^T(x)C^{-1}(x)YA(x)\right)s\right\rangle, \tag{6.9}$$

possibly subject to a trust-region constraint. We have already noticed that the first-order term $g(x,y(x,\mu)) = \nabla_x\Phi(x,\mu)$ as $y(x,\mu)=\mu C^{-1}(x)e$, and thus the model gradient is that of the barrier function as required by our global convergence analyses of linesearch and trust-region methods. We have discounted always choosing $y = y(x,\mu)$ in (6.9), and have suggested that the choice $y = (\mu_{k-1}/\mu_k)y(x,\mu)$ when changing the barrier parameter results in good use of the starting point. Another possibility is to use $y = y^{\text{OLD}} + w^{\text{OLD}}$, where w^{OLD} is the primal-dual correction to the previous dual-variable estimates y^{OLD}. However, this needs to be used with care since there is no a priori assurance that $y^{\text{OLD}}+w^{\text{OLD}}>0$, and indeed it is usual to prefer $y=\max(y^{\text{OLD}}+w^{\text{OLD}},\epsilon(\mu_k)e)$ for some "small" $\epsilon(\mu_k)>0$. The choice $\epsilon(\mu_k)=\mu_k^{1.5}$ leads to a realistic primal-dual method, although other precautions need sometimes to be taken.

6.4.3 Potential difficulty I . . . revisited

We now return to the other perceived difficult with barrier or primal-dual path-following methods, namely that the inherent ill-conditioning in the barrier Hessian makes it hard to generate accurate Newton steps when the barrier parameter is small. Let $\mathcal{I}$ be the set of inactive constraints at x_*, and denote the active and inactive components of c and y with suffices $\mathcal{A}$ and $\mathcal{I}$ respectively. Thus $c_{\mathcal{A}}(x_*)=0$ and $c_{\mathcal{I}}(x_*)>0$, while if the solution is non-degenerate, $y_{\mathcal{A}}(x_*)>0$ and $y_{\mathcal{I}}(x_*)=0$. As we have seen, the Newton correction s^{PD} satisfies (6.7), while the equivalent system (6.8) clearly has a condition number that approaches infinity as x and y reach their limits because $c_{\mathcal{A}}(x)$ approaches zero while $y_{\mathcal{A}}(x)$ approaches $y_{\mathcal{A}}(x_*)>0$.

But now suppose that we separate (6.7) into

$$\begin{pmatrix} H(x,y) & -A_{\mathcal{A}}^T(x) & -A_{\mathcal{I}}^T(x) \\ Y_{\mathcal{A}}A_{\mathcal{A}}(x) & C_{\mathcal{A}}(x) & 0 \\ Y_{\mathcal{I}}A_{\mathcal{A}}(x) & 0 & C_{\mathcal{I}}(x) \end{pmatrix} \begin{pmatrix} s^{\text{PD}} \\ w_{\mathcal{A}} \\ w_{\mathcal{I}} \end{pmatrix} = -\begin{pmatrix} g(x)-A^T(x)y \\ C_{\mathcal{A}}(x)y_{\mathcal{A}}-\mu e \\ C_{\mathcal{I}}(x)y_{\mathcal{I}}-\mu e \end{pmatrix},$$

and then eliminate the variables $w_{\mathcal{I}}$, multiply the second equation by $Y_{\mathcal{A}}^{-1}$ and use $C_{\mathcal{I}}(x)y_{\mathcal{I}}=\mu e$, we obtain

$$\begin{aligned}\begin{pmatrix} H(x,y)+A_{\mathcal{I}}^T(x)C_{\mathcal{I}}(x)^{-1}Y_{\mathcal{I}}A_{\mathcal{I}}(x) & -A_{\mathcal{A}}^T(x) \\ A_{\mathcal{A}}(x) & C_{\mathcal{A}}(x)Y_{\mathcal{A}}^{-1} \end{pmatrix} \begin{pmatrix} s^{\text{PD}} \\ w_{\mathcal{A}} \end{pmatrix} \\ = -\begin{pmatrix} g(x)-A_{\mathcal{A}}^T(x)y_{\mathcal{A}}-\mu A_{\mathcal{I}}^T(x)C_{\mathcal{I}}^{-1}(x)e \\ c_{\mathcal{A}}(x)-\mu Y_{\mathcal{A}}^{-1}e \end{pmatrix}.\end{aligned} \tag{6.10}$$

But then we see that the terms involving inverses, $C_{\mathcal{I}}^{-1}(x)$ and $Y_{\mathcal{A}}^{-1}$, remain bounded, and indeed in the limit the system becomes

$$\begin{pmatrix} H(x,y) & -A_{\mathcal{A}}^T(x) \\ A_{\mathcal{A}}(x) & 0 \end{pmatrix} \begin{pmatrix} s^{\text{PD}} \\ w_{\mathcal{A}} \end{pmatrix} = -\begin{pmatrix} g(x)-A_{\mathcal{A}}^T(x)y_{\mathcal{A}}-\mu A_{\mathcal{I}}^T(x)C_{\mathcal{I}}^{-1}(x)e \\ 0 \end{pmatrix}$$

which is well behaved. Thus just because (6.8) is ill conditioned, this does not preclude us from finding s^{PD} from an equivalent, perfectly well-behaved system like (6.10).

6.5 A practical primal-dual method

Following on from the above, we now give the skeleton of a reasonable primal-dual method.

> Given $\mu_0 > 0$ and feasible $(x_0^{\mathrm{S}}, y_0^{\mathrm{S}})$, set $k = 0$.
> Until "convergence", iterate:
> *Inner minimization*: starting from $(x_k^{\mathrm{S}}, y_k^{\mathrm{S}})$, use an unconstrained minimization algorithm to find (x_k, y_k) for which
> $\|C(x_k)y_k - \mu_k e\| \leq \mu_k$ and $\|g(x_k) - A^T(x_k)y_k\| \leq \mu_k^{1.00005}$.
> Set $\mu_{k+1} = \min(0.1\mu_k, \mu_k^{1.9999})$.
> Find $(x_{k+1}^{\mathrm{S}}, y_{k+1}^{\mathrm{S}})$ using a primal-dual Newton step from (x_k, y_k).
> If $(x_{k+1}^{\mathrm{S}}, y_{k+1}^{\mathrm{S}})$ is infeasible, reset $(x_{k+1}^{\mathrm{S}}, y_{k+1}^{\mathrm{S}})$ to (x_k, y_k).
> Increase k by 1.

The inner minimization will be performed by either a linesearch or trust-region method for minimizing $\Phi(x, \mu_k)$, the stopping rules $\|C(x_k)y_k - \mu_k e\| \leq \mu_k$ and $\|g(x_k) - A^T(x_k)y_k\| \leq \mu_k^{1.00005}$ certainly being attainable as the first-order optimality condition for minimizing $\Phi(x, \mu_k)$ is that $g(x) - A^T(x)y = 0$, where $C(x)y = \mu_k e$. The extra step, in which the starting point is computed by performing a primal-dual Newton step from (x_k, y_k), is simply included to generate a value that is already close to first order critical, and the stopping tolerances are specially chosen to encourage this. Indeed we have the following asymptotic convergence result.

> **Theorem 6.3.** Suppose that f, $c \in \mathcal{C}^2$, that a subsequence $\{(x_k, y_k)\}$, $k \in \mathcal{K}$, of the practical primal-dual method converges to (x_*, y_*) satisfying second-order sufficiency conditions, that $A_{\mathcal{A}}(x_*)$ is full-rank, and that $(y_*)_{\mathcal{A}} > 0$. Then the starting point satisfies the inner-minimization termination test (i.e., $(x_k, y_k) = (x_k^{\mathrm{S}}, y_k^{\mathrm{S}})$) for all k sufficiently large, and the whole sequence $\{(x_k, y_k)\}$ converges to (x_*, y_*) at a superlinear rate (with a Q-factor at least 1.9998).

This is a highly acceptable result, the convergence being essentially quadratic (which would correspond to a Q-factor of two—any sequence $\{\sigma_k\}$ is said to converge to σ_* with *Q-factor* at least q if $|\sigma_{k+1} - \sigma_*| \leq \gamma|\sigma_k - \sigma_*|^q$ for some $\gamma > 0$).

Primal-dual interior-point methods have the potential for both excellent theoretical and practical behaviour. There are polynomial interior-point algorithms for linear, (convex) quadratic and semi-definite programming. While it is unlikely that this is true for more general (nonconvex) problems, the barrier function globalization is most effective in practice, and the asymptotic behaviour is normally just as for the convex case. From a global perspective, it is very important that iterates are kept away from constraint boundary until near to convergence, as otherwise very slow progress will be made—this is certainly born out in practice. Finally, while the methods we have discussed in this parthave all required an interior starting point, it is possible to find one (if there is one!) by solving the "phase-one" problem to

$$\underset{(x,\gamma)}{\text{minimize}} \;\; \gamma \;\text{ subject to }\; c(x) + \gamma e \geq 0;$$

any feasible point (x, γ) for this auxiliary problem for which $\gamma < 0$ is suitable, for then $c(x) > 0$.

It is quite common in practice to replace the inequality $c_i(x) \geq 0$ by the equation $c_i(x) - s_i = 0$, and simple bound $s_i \geq 0$ on the *slack* variable s_i. This has the algebraic advantage that the inequality constraints are then all simple bounds and thus that barrier terms only appear on the diagonal of the Hessian model, but arguably the disadvantages that the dimensionality of the problem has been artificially increased, and that we now need to use some means of coping with equality constraints. We consider this latter point next.

PART 7

SQP METHODS FOR EQUALITY CONSTRAINED OPTIMIZATION

In this final part, having already investigated very good methods for dealing with inequality constraints, we now turn our attention to the problem (5.1), in which there are only equality constraints on the variables. Of course in practice, there are frequently both equations and inequalities, and composite methods using the barrier/interior-point methods discussed in Part 6 and the SQP methods we shall consider here are often used. Alternatively, SQP methods themselves may easily be generalized to handle inequality constraints. For brevity we shall not consider such extensions further here.

7.1 Newton's method for first-order optimality

Sequential Quadratic Programming (SQP) methods (sometimes called successive or recursive quadratic programming methods) are most naturally derived by considering the first-order necessary conditions for (5.1)—we will see where the names come from shortly. Recall at optimality we expect to have

$$g(x,y) \equiv g(x) - A^T(x)y = 0 \;\text{ and }\; c(x) = 0. \tag{7.1}$$

This is a system of nonlinear equations in the variables x and the Lagrange multipliers y. Notice that the system is actually linear in y so that if x were known it would be straightforward to find y.

Suppose now that (x,y) is an approximation to a solution of (7.1). Then, as always, we might apply Newton's method to try to improve (x,y), and this leads us to construct a correction (s,w) for which

$$\begin{pmatrix} H(x,y) & -A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ w \end{pmatrix} = - \begin{pmatrix} g(x,y) \\ c(x) \end{pmatrix}. \tag{7.2}$$

Newton's method would then apply the same procedure to the "improved" estimate $(x_+, y_+) = (x+s, y+w)$.

There are a number of alternative formulations of (7.2). Firstly (7.2) may be written as the symmetric system of equations

$$\begin{pmatrix} H(x,y) & A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ -w \end{pmatrix} = - \begin{pmatrix} g(x,y) \\ c(x) \end{pmatrix};$$

notice here that the coefficient matrix is indefinite because of its zero 2,2 block. Secondly, on writing $y_+ = y + w$, the equation becomes

$$\begin{pmatrix} H(x,y) & -A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ y_+ \end{pmatrix} = - \begin{pmatrix} g(x) \\ c(x) \end{pmatrix},$$

or finally, in symmetric form,

$$\begin{pmatrix} H(x,y) & A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ -y_+ \end{pmatrix} = - \begin{pmatrix} g(x) \\ c(x) \end{pmatrix}.$$

In practice we might prefer to approximate $H(x,y)$ by some symmetric B, and instead solve

$$\begin{pmatrix} B & A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ -y_+ \end{pmatrix} = - \begin{pmatrix} g(x) \\ c(x) \end{pmatrix} = \begin{pmatrix} B & -A^T(x) \\ A(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ y_+ \end{pmatrix}. \tag{7.3}$$

One could imagine solving these related systems by finding an LU factorization of the coefficient matrix in the unsymmetric case, or a symmetric-indefinite (a generalization of Cholesky) factorization in the symmetric case. Alternatively, if B is invertible, s and y_+ might be found successively by solving

$$A(x)B^{-1}A(x)^T y = -c + A(x)B^{-1}g \text{ and then } Bs = A(x)^T y - g$$

using symmetric factorizations of B and $A(x)B^{-1}A(x)^T$ (see Section 4.3.1). For very large problems, iterative methods might be preferred, and here GMRES(k) or QMR, for the unsymmetric case, or MINRES or conjugate-gradients (restricted to the null-space of $A(x)$), for the symmetric case, have all been suggested. Thus there are many ways to solve the system(s) of linear equations that arise from SQP methods, and there is currently much interest in exploiting the structure in such systems to derive very efficient methods.

But where does the name "sequential quadratic programming" come from?

7.2 The Sequential Quadratic Programming iteration

As we saw in Part 4, a quadratic program is a problem involving the optimization of a quadratic function subject to a set of linear inequality and/or equality constraints. Consider the quadratic programming problem

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \;\; \langle s, g(x) \rangle + \tfrac{1}{2}\langle s, Bs \rangle \text{ subject to } A(x)s = -c(x). \tag{7.4}$$

Why this problem? Well, Theorem 1.3 indicates that $c(x) + A(x)s$ is a first-order (Taylor) approximation to the constraint function $c(x+s)$, while $\langle s, g(x) \rangle + \frac{1}{2}\langle s, Bs \rangle$ is potentially a second-order model of the decrease $f(x+s) - f(x)$. Thus one can argue that (7.4) gives a suitable (at least first-order) model of (5.1). An objection might be that really we should be aiming for true second-order approximations to all functions concerned, but this would lead to the significantly-harder minimization of a quadratic function subject to quadratic constraints—constraint curvature is a major obstacle.

The interesting feature of (7.4) is that it follows immediately from Theorem 1.7 that any first-order critical point of (7.4) is given by (7.3). Thus Newton-like methods for first-order optimality are equivalent to the solution of a sequence of related quadratic programs. Hence the name. Notice that if $B = H(x, y)$, solving (7.4) is actually Newton's method for (7.1), and this suggests that B should be an approximation to the Hessian of the Lagrangian function, not the objective function. Clearly the constraint curvature that we would have liked to have added to the linear approximations of the constraints has worked its way into the objective function!

To summarize, the basic SQP iteration is as follows.

Given (x_0, y_0), set $k = 0$.
Until "convergence" iterate:
    Compute a suitable symmetric B_k using (x_k, y_k).
    Find

$$s_k = \arg\min_{s \in \mathbb{R}^n} \langle s, g_k \rangle + \tfrac{1}{2}\langle s, B_k s \rangle \text{ subject to } A_k s = -c_k \tag{7.5}$$

    along with associated Lagrange multiplier estimates y_{k+1}.
    Set $x_{k+1} = x_k + s_k$ and increase k by 1.

The SQP method is both simple and fast. If $B_k = H(x_k, y_k)$, the method is Newton's method for (7.1), and thus is quadratically convergent provided that (x_0, y_0) is sufficiently close to a first-order critical point (x_*, y_*) of (5.1) for which

$$\begin{pmatrix} H(x_*, y_*) & A^T(x_*) \\ A(x_*) & 0 \end{pmatrix}$$

is non-singular. Moreover, the method is superlinearly convergent when B_k is a "good" approximation to $H(x_k, y_k)$, and there is even no necessity that this be so for fast convergence. It should also be easy for the reader to believe that had we wanted to solve the problem (6.1) involving inequality constraints, the suitable SQP subproblem would be

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \;\; \langle s, g(x) \rangle + \tfrac{1}{2}\langle s, Bs \rangle \text{ subject to } A(x)s \geq -c(x)$$

in which the nonlinear inequalities have been linearized.

But, as the reader will already have guessed, this basic iteration also has drawbacks, leading to a number of vital questions. For a start it is a Newton-like iteration, and thus may diverge from poor starting points. So how do we globalize this iteration? How should we pick B_k? What should we do if (7.4) is unbounded from below? And precisely when is it unbounded?

The problem (7.4) only has a solution if the constraints $A(x)s = -c(x)$ are consistent. This is certainly the case if $A(x)$ is full rank, but may not be so if $A(x)$ is rank deficient—we shall consider alternatives that deal with this deficiency later. Applying Theorem 1.8 to (7.4), we deduce that any stationary point (s, y_+) satisfying (7.3) solves (7.4) only if B is positive semi-definite on the manifold $\{s : A(x)s = 0\}$—if B is positive definite on the manifold (s, y_+) is the unique solution to the problem. If the m by n matrix $A(x)$ is full rank and the columns of $S(x)$ form a basis for the null-space of $A(x)$, it is easy to show that B being positive (semi-)definite on the manifold $\{s : A(x)s = 0\}$ is equivalent to $S(x)^T BS(x)$ being positive (semi-)definite which is in turn equivalent to the matrix

$$\begin{pmatrix} B & A^T(x) \\ A(x) & 0 \end{pmatrix}$$

(being non-singular and) having m negative eigenvalues. If B violates these assumptions, (7.4) is unbounded.

For the remainder of this part, we focus on methods to globalize the SQP iteration. And it should not surprise the reader that we shall do so by considering linesearch and trust-region schemes.

7.3 Linesearch SQP methods

The obvious way to embed the SQP step s_k within a linesearch framework is to pick $x_{k+1} = x_k + \alpha_k s_k$, where the step $\alpha_k > 0$ is chosen so that

$$\Phi(x_k + \alpha_k s_k, p_k) \text{ “}<\text{” } \Phi(x_k, p_k), \tag{7.6}$$

and where $\Phi(x, p)$ is a “suitable” merit function depending on parameters p_k. Of course it is then vital that s_k be a descent direction for $\Phi(x, p_k)$ at x_k, as otherwise there may be no α_k for which (7.6) is satisfied. As always with linesearch methods, this limits the choice of B_k, and it is usual to insist that B_k be positive definite—the reader may immediately object that this is imposing an unnatural requirement, since B_k is supposed to be approximating the (usually) indefinite matrix $H(x_k, y_k)$, and we can only sympathise with such a view!

What might a suitable merit function be? One possibility is to use the quadratic penalty function (5.2). In this case, we have the following result.

Theorem 7.1. Suppose that B_k is positive definite, and that (s_k, y_{k+1}) are the SQP search direction and its associated Lagrange multiplier estimates for the problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x) \;\text{ subject to }\; c(x) = 0$$

at x_k. Then if x_k is not a first-order critical point, s_k is a descent direction for the quadratic penalty function $\Phi(x, \mu_k)$ at x_k whenever

$$\mu_k \leq \frac{\|c(x_k)\|_2}{\|y_{k+1}\|_2}.$$

We know that the parameter μ_k for the quadratic penalty function needs to approach zero for its minimizers to converge to those of (5.1), so Theorem 7.1 simply confirms this by suggesting how to adjust the parameter.

The quadratic penalty function has another role to play if the constraints are inconsistent. For consider the quadratic (Newton-like) model

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \;\; \langle s, g_k + A_k^T c_k/\mu_k \rangle + \tfrac{1}{2} \langle s, (B_k + 1/\mu_k A_k^T A_k) s \rangle$$

that might be used to compute a step s_k^{Q} from x_k. Stationary points of this model satisfy

$$(B_k + 1/\mu_k A_k^T A_k) s_k^{\mathrm{Q}} = -(g_k + A_k^T c_k/\mu_k)$$

or, on defining $y_k^{\mathrm{Q}} \stackrel{\mathrm{def}}{=} -\mu_k^{-1}(c_k + A_k s_k^{\mathrm{Q}})$,

$$\begin{pmatrix} B_k & A_k^T \\ A_k & -\mu_k I \end{pmatrix} \begin{pmatrix} s_k^{\mathrm{Q}} \\ -y_k^{\mathrm{Q}} \end{pmatrix} = - \begin{pmatrix} g_k \\ c_k \end{pmatrix}. \tag{7.7}$$

But now compare this system with (7.3) that which defines the SQP step: the only difference is the vanishingly small 2,2 block $-\mu_k I$ in the coefficient matrix. While this indicates that Newton-like directions for the quadratic penalty function will become increasingly good approximations to SQP steps (and, incidentally, it can be shown that a Newton iteration for (5.2) with well chosen μ_k converges superlinearly under reasonable assumptions), the main point of the alternative (7.7) is that rank-deficiency in A_k is neutralised by the presence of 2,2 block term $-\mu_k I$. Nevertheless, the quadratic penalty function is rarely used, its place often being taken by non-differentiable exact penalty functions.

The *non-differentiable exact penalty function* is given by

$$\Phi(x, \rho) = f(x) + \rho \|c(x)\| \tag{7.8}$$

for any norm $\|\cdot\|$ and scalar $\rho > 0$. Notice that the function is non-differentiable particularly when $c(x) = 0$, the very values we hope to attain! The following result helps explain why such a function is considered so valuable.

Theorem 7.2. Suppose that $f, c \in C^2$, and that x_* is an isolated local minimizer of $f(x)$ subject to $c(x) = 0$, with corresponding Lagrange multipliers y_*. Then x_* is also an isolated local minimizer of $\Phi(x, \rho)$ provided that

$$\rho > \|y_*\|_D,$$

where the *dual norm*

$$\|y\|_D = \sup_{x \neq 0} \frac{\langle y, x \rangle}{\|x\|}.$$

Notice that the fact that ρ merely needs to be larger than some critical value for $\Phi(x, \rho)$ to be usable to try to identify solutions to (5.1) is completely different to the quadratic penalty function, for which the parameter had to take on a limiting value.

More importantly, as we now see, $\Phi(x, \rho)$ may be used as a merit function for the SQP step.

Theorem 7.3. Suppose that B_k is positive definite, and that (s_k, y_{k+1}) are the SQP search direction and its associated Lagrange multiplier estimates for the problem

$$\operatorname*{minimize}_{x \in \mathbb{R}^n} \;\; f(x) \;\; \text{subject to} \;\; c(x) = 0$$

at x_k. Then if x_k is not a first-order critical point, s_k is a descent direction for the non-differentiable penalty function $\Phi(x, \rho_k)$ at x_k whenever $\rho_k \geq \|y_{k+1}\|_D$.

Once again, this theorem indicates how ρ_k needs to be adjusted for use within a linesearch SQP framework.

Thus far, everything looks perfect. We have methods for globalizing the SQP iteration, an iteration that should ultimately converge very fast. But unfortunately, it is not as simple as that. For consider the example in Figure 7.1. Here the current iterate lies close to (actually on) the

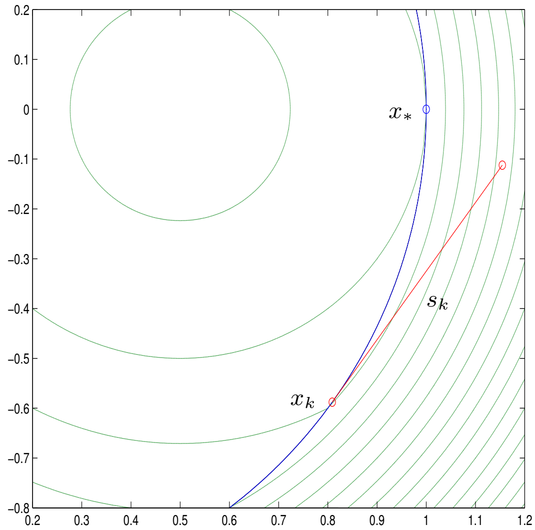

Figure 7.1: ℓ_1 non-differentiable exact penalty function ($\rho = 1$): $f(x) = 2(x_1^2 + x_2^2 - 1) - x_1$ and $c(x) = x_1^2 + x_2^2 - 1$. Solution: $x_* = (1, 0)$, $y_* = \frac{3}{2}$. The SQP direction using the optimal Hessian $H(x_*, y_*) = I$. Notice how the merit function increases at the point $x_k + s_k$.

constraint, the SQP step moves tangentially from it, and thus moves away as the constraint is nonlinear, but unfortunately, at the same time, the value of the objective function rises. Thus any merit function like (5.2) or (7.8) composed simply from positive combinations of the objective and (powers) of norms of constraint violations will increase after such an SQP step, and thus necessarily $\alpha_k \neq 1$ in (7.6)—worse still, this behaviour can happen arbitrarily close to the minimizer. This has the unfortunate side effect that it may happen that the expected fast convergence achievable by Newton-like methods will be thwarted by the merit function. That is, there is a serious mismatch between the global and local convergence needs of the SQP method. The fact that the merit function may prevent acceptance of the full SQP step is known as the *Maratos effect*.

The Maratos effect occurs because the curvature of the constraints is not adequately represented by linearization in the SQP model. In particular,

$$c(x_k + s_k) = O(\|s_k\|^2).$$

This suggests that we need to correct for this curvature. We may do this by computing a *second-order correction* from $x_k + s_k$, that is an extra step s_k^{C} for which

$$c(x_k + s_k + s_k^{\mathrm{C}}) = o(\|s_k\|^2). \tag{7.9}$$

Since we do not want to destroy potential for fast convergence, we must also insist that the correction is small relative to the SQP step, and thus that

$$s_k^{\mathrm{C}} = o(s_k). \tag{7.10}$$

There are a number of ways to compute a second-order correction. The first is simply to move back as quickly as possible towards the constraints. This suggests we compute a minimum (ℓ_2-)norm solution to $c(x_k + s_k) + A(x_k + s_k)s_k^{\mathrm{C}} = 0$. It is easy to check that the required solution satisfies

$$\begin{pmatrix} I & A^T(x_k + s_k) \\ A(x_k + s_k) & 0 \end{pmatrix} \begin{pmatrix} s_k^{\mathrm{C}} \\ -y_{k+1}^{\mathrm{C}} \end{pmatrix} = - \begin{pmatrix} 0 \\ c(x_k + s_k) \end{pmatrix}.$$

Since this requires that we re-evaluate the constraints *and* their Jacobian at $x_k + s_k$, we might hope instead to find a minimum norm solution to $c(x_k + s_k) + A(x_k)s_k^{\mathrm{C}} = 0$, and thus that

$$\begin{pmatrix} I & A^T(x_k) \\ A(x_k) & 0 \end{pmatrix} \begin{pmatrix} s_k^{\mathrm{C}} \\ -y_{k+1}^{\mathrm{C}} \end{pmatrix} = - \begin{pmatrix} 0 \\ c(x_k + s_k) \end{pmatrix}.$$

A third amongst many other possibilities is to compute another SQP step from $x_k + s_k$, that is to compute s_k^{C} so that

$$\begin{pmatrix} B_k^{\mathrm{C}} & A^T(x_k + s_k) \\ A(x_k + s_k) & 0 \end{pmatrix} \begin{pmatrix} s_k^{\mathrm{C}} \\ -y_{k+1}^{\mathrm{C}} \end{pmatrix} = - \begin{pmatrix} g(x_k + s_k) \\ c(x_k + s_k) \end{pmatrix},$$

where B_k^{C} is an approximation to $H(x_k + s_k, y_k^+)$. It can easily be shown that all of the above corrections satisfy (7.9)–(7.10). In Figure 7.2, we illustrate a second-order correction in action. It is possible to show that, under reasonable assumptions, any step $x_k + s_k + s_k^{\mathrm{C}}$ made up from the SQP step s_k and a second-order correction s_k^{C} satisfying (7.9)–(7.10) will ultimately reduce (7.8). So now we can have both global and very fast asymptotic convergence at the expense of extra problem evaluations. Of course, we have stressed that a second SQP step gives a second-order correction, so another way of viewing this is to require that the merit function decreases at least every second iteration, and to tolerate non-monotonic behaviour in the interim.

7.4 Trust-region SQP methods

The main disadvantage of (at least naive) linesearch SQP methods is the unnatural requirement that B_k be positive definite. We saw the same restriction in the unconstrained case, although at least then there was some expectation that ultimately the true Hessian H_k would be positive (semi-)definite. In the unconstrained case, indefinite model Hessians were better handled in a trust-region framework, and the same is true in the constrained case.

The obvious trust-region generalization of the basic SQP step-generation subproblem (7.4) is to find

$$s_k = \underset{s \in \mathbb{R}^n}{\arg\min} \ \langle s, g_k \rangle + \tfrac{1}{2}\langle s, B_k s \rangle \ \text{ subject to } \ A_k s = -c_k \ \text{ and } \ \|s\| \leq \Delta_k. \tag{7.11}$$

Since we do not require that B_k be positive definite, this allows us to use $B_k = H(x_k, y_k)$ if we so desire. However a few moments reflection should make it clear that such an approach has a serious flaw. Let Δ^{CRIT} be the least distance to the linearized constraints, i.e.

$$\Delta^{\mathrm{CRIT}} \stackrel{\text{def}}{=} \min \|s\| \ \text{ subject to } \ A_k s = -c_k.$$

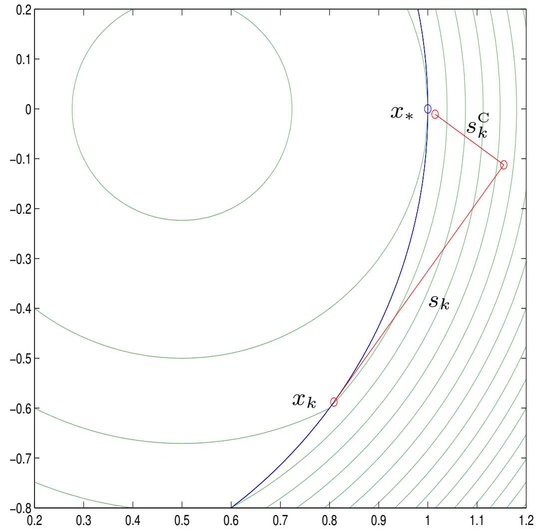

Figure 7.2: ℓ_1 non-differentiable exact penalty function ($\rho = 1$): $f(x) = 2(x_1^2 + x_2^2 - 1) - x_1$ and $c(x) = x_1^2 + x_2^2 - 1$ solution: $x_* = (1, 0)$, $y_* = \frac{3}{2}$. See that the second-order correction s_k^C helps avoid the Maratos effect for the above problem with the ℓ_1-penalty function. Notice how s_k^C more than compensates for the increase in the merit function at the point $x_k + s_k$, and how much closer $x_k + s_k + s_k^C$ is to x_* than is x_k.

The difficulty is that if $\Delta_k < \Delta^{\text{CRIT}}$, then there is *no solution* to the trust-region subproblem (7.11). This implies that unless $c_k = 0$, the subproblem is meaningless for all sufficiently small trust-region radius (see Figure 7.3). Thus we need to consider alternatives. In this section, we shall review the S$\ell_{\mathbf{p}}$QP method of Fletcher, the composite step SQP methods due to Vardi, to Byrd and Omojokun, and to Celis, Dennis and-Tapia, and the filter-SQP approach of Fletcher and Leyffer.

7.4.1 The S$\ell_{\mathbf{p}}$QP method

Our first trust-region approach is to try to minimize the ℓ_p-(exact) penalty function

$$\Phi(x, \rho) = f(x) + \rho \|c(x)\|_p \tag{7.12}$$

for sufficiently large $\rho > 0$ and some ℓ_p norm ($1 \leq p \leq \infty$). We saw in Part 7.3 that feasible minimizers of (7.12) may be solutions to (5.1) so long as $\rho > 0$ is large enough. Of course, as $\Phi(x, \rho)$ is non-differentiable, we cannot simply apply one of the unconstrained trust-region methods discussed in Part 3, but must instead build a specialized method.

Since we are discussing trust-region methods, a suitable model problem is the *$\ell_{\mathbf{p}}$QP*

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \quad f_k + \langle s, g_k \rangle + \tfrac{1}{2}\langle s, B_k s \rangle + \rho \|c_k + A_k s\|_p \quad \text{subject to} \quad \|s\| \leq \Delta_k.$$

This has the major advantage that the model problem is always consistent, since now the only constraint is the trust-region bound. In addition, when ρ and Δ_k are large enough, it can be shown that the model minimizer *is* the SQP direction so long as $A_k s = -c_k$ is consistent. Moreover, when the norms are polyhedral (e.g., the ℓ_1 or ℓ_∞ norms), $\ell_{\mathbf{p}}$QP is equivalent to a quadratic program.

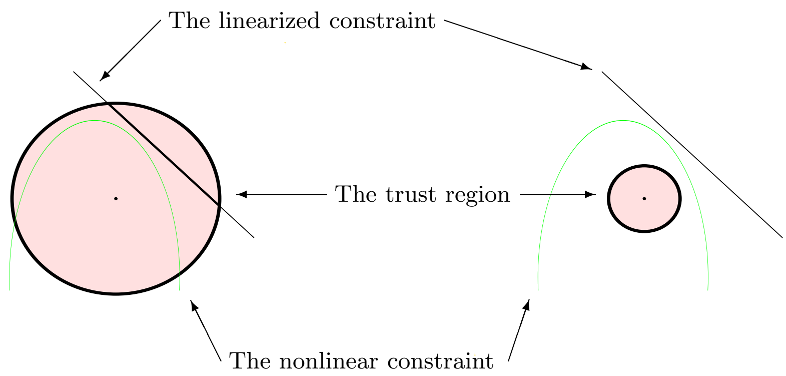

Figure 7.3: The intersection between the linearization of a nonlinear constraint and a spherical trust region. In the left figure, the trust-region radius is sufficiently large for the trust region and the linearized constraint to intersect. This is not so for the smaller trust region illustrated in the right figure.

To see this, consider for example the $\ell_{\mathbf{1}}$QP model problem with an ℓ_∞ trust region

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; \langle s, g_k\rangle + \tfrac{1}{2}\langle s, B_k s\rangle + \rho\|c_k + A_k s\|_1 \;\; \text{subject to} \;\; \|s\|_\infty \leq \Delta_k.$$

But we can always write

$$c_k + A_k s = u - v, \;\; \text{where} \;\; (u, v) \geq 0.$$

Hence the $\ell_{\mathbf{1}}$QP subproblem is equivalent to the quadratic program

$$\begin{array}{rl} \underset{s\in\mathbb{R}^n,\; u,v\in\mathbb{R}^m}{\text{minimize}} & \langle s, g_k\rangle + \frac{1}{2}\langle s, B_k s\rangle + \rho\langle e, u+v\rangle \\ \text{subject to} & A_k s - u + v = -c_k \\ & u \geq 0, \;\; v \geq 0 \\ \text{and} & -\Delta_k e \leq s \leq \Delta_k e. \end{array}$$

Notice that the QP involves inequality constraints, but there are good methods (especially of the interior-point variety) for solving such problems. In particular, it is possible to exploit the structure of the u and v variables.

In order to develop a practical S$\ell_{\mathbf{1}}$QP method, it should not surprise the reader that we need to ensure that every step we generate achieves as much reduction in the model $f_k + \langle s, g_k\rangle + \frac{1}{2}\langle s, B_k s\rangle + \rho\|c_k + A_k s\|_p$ as would have been achieved at a Cauchy point. One such Cauchy point requires the solution to $\ell_{\mathbf{1}}$LP model

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; \langle s, g_k\rangle + \rho\|c_k + A_k s\|_1 \;\; \text{subject to} \;\; \|s\|_\infty \leq \Delta_k,$$

which may be reformulated as a linear program. Fortunately approximate solutions to both $\ell_{\mathbf{1}}$LP and $\ell_{\mathbf{1}}$QP subproblems suffice. In practice it is also important to adjust ρ as the method progresses so as to ensure that ρ is larger than the (as yet unknown) $\|y_*\|_D$, and this may be achieved by using the available Lagrange multiplier estimates y_k. Such a scheme is globally convergent, but there is still a need for a second-order correction to prevent the Maratos effect and thus allow fast asymptotic convergence. If $c(x) = 0$ are inconsistent, the method converges to (locally) least value of the infeasibility $\|c(x)\|$ provided $\rho \to \infty$.

The alert reader will have noticed that in this section we have replaced the ℓ_2 trust-region of the unconstraint trust-region method by a box or ℓ_∞ trust-region. The reason for this apparent lack of consistency is that minimizing a quadratic subject to linear constraints *and* an additional quadratic trust-region is too hard. On the other hand, adding box-constraints does not increase the complexity of the resulting (quadratic programming) trust-region subproblem.

7.4.2 Composite-step methods

Another approach to avoid the difficulties caused by inconsistent QP subproblems is to separate the computation of the step into two stages. The aim of a *composite-step* method is to find

$$s_k = n_k + t_k,$$

where the *normal step* n_k moves towards feasibility of the linearized constraints (within the trust region), while the *tangential step* t_k reduces the model objective function (again within the trust-region) without sacrificing feasibility obtained from n_k. Of course since the normal step is solely concerned with feasibility, the model objective may get worse, and indeed it may not recover during the tangential step. The fact that the tangential step is required to maintain any gains in (linearized) feasibility achieved during the normal step implies that

$$A_k(n_k + t_k) = A_k n_k \;\text{ and hence that }\; A_k t_k = 0.$$

We illustrate possible normal and tangential steps in Figure 7.4.

7.4.2.1 Constraint relaxation—Vardi's method

Vardi's approach is an early composite-step method. The normal step is found by relaxing the requirement

$$A_k s = -c_k \;\text{ and }\; \|s\| \le \Delta_k$$

to

$$A_k n = -\sigma_k c_k \;\text{ and }\; \|n\| \le \Delta_k,$$

where $\sigma_k \in [0, 1]$ is small enough so that there is a feasible n_k. Clearly $s = 0$ is feasible if $\sigma_k = 0$, and the largest possible $\sigma_{\max}$ may be found by computing

$$\max_{\sigma \in (0,1]} \left[\min_{\|s\| \le \Delta_k} \|A_k s + \sigma c_k\| = 0 \right].$$

In practice, some value between zero and $\sigma_{\max}$ is chosen, since this gives some "elbow-room" in which to compute the tangential step. The main defect with the approach is that there may be no normal step if the linearized constraints are inconsistent.

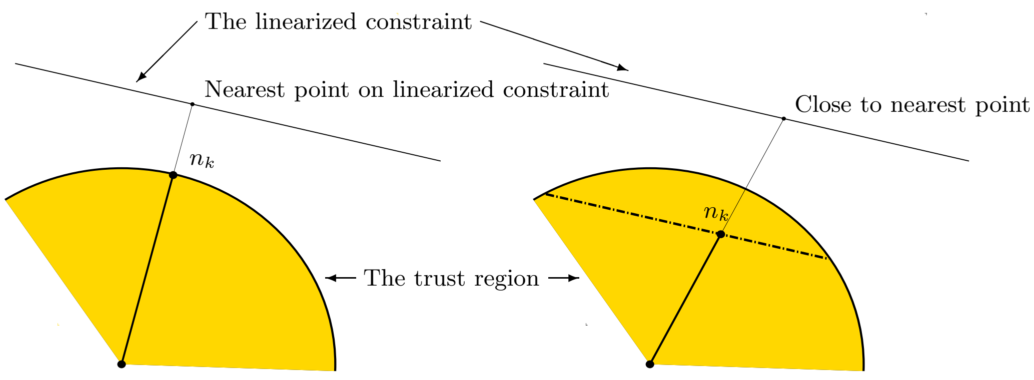

Figure 7.4: Computing the normal step. The left-hand figure shows the largest possible normal step. The right-hand figure illustrates a shorter normal step n, and the freedom this then allows for the tangential step—any point on the dotted line is a potential tangential step.

Once a normal step has been determined, the tangential step is computed as the

$$\begin{array}{rl}(\text{approximate})\ \underset{t \in \mathbb{R}^n}{\arg\min} & \langle t, g_k + B_k n_k \rangle + \tfrac{1}{2}\langle t, B_k t \rangle \\ \text{subject to} & A_k t = 0 \ \text{ and } \ \|n_k + t\| \leq \Delta_k.\end{array}$$

Although of historical interest, the method has been effectively superseded by the Byrd–Omojokun approach we describe next.

7.4.2.2 Constraint reduction—the Byrd–Omojokun method

The Byrd–Omojokun method aims to cope with the inconsistency issue that afflicts Vardi's approach. Rather than relaxing the constraints, the normal step is now computed as

$$\text{approximately minimize } \ \|A_k n + c_k\| \ \text{ subject to } \ \|n\| \leq \Delta_k,$$

in order to achieve a reasonable improvement in linearized infeasibility that is consistent with the trust-region. The tangential step is then computed exactly as in Vardi's method.

An important aspect is that it is possible to use the conjugate gradient method to solve both subproblems. This provides Cauchy points in both cases and allows the method to be used to solve large problems. The method has been shown to be globally convergent (under reasonable assumptions) using an ℓ_2 merit function, and is the basis of the successful `KNITRO` software package.

7.4.2.3 Constraint lumping—the Celis–Dennis–Tapia method

A third method which might be considered to be of the composite-step variety is that due to Celis, Dennis and Tapia. In this approach, the requirement that $A_k s = -c_k$ is replaced by requiring that

$$\|A_k s + c_k\| \leq \sigma_k$$

for some $\sigma_k \in [0, \|c_k\|]$. The value of σ_k is chosen so that the normal step n_k satisfies

$$\|A_k n + c_k\| \leq \sigma_k \text{ and } \|n\| \leq \Delta_k.$$

Having found a suitable normal step, the tangential step is found as an

$$\begin{array}{ll}\text{(approximate) } \underset{t \in \mathbb{R}^n}{\arg\min} & \langle t, g_k + B_k n_k \rangle + \frac{1}{2}\langle t, B_k t \rangle \\ \text{subject to} & \|A_k t + A_k n_k + c_k\| \leq \sigma_k \text{ and } \|t + n_k\| \leq \Delta_k.\end{array}$$

While finding a suitable σ_k is inconvenient, the real Achilles' heel of this approach is that the tangential step subproblem is (much) harder than those we have considered so far. If the ℓ_2-norm is used for the constraints, we need to find the minimizer of a quadratic objective within the intersection of two "spherical" regions. Unlike the case involving a single sphere (recall Part 3.5.1), it is not known if there is an efficient algorithm in the two-sphere case. Alternatively, if polyhedral (ℓ_1 or ℓ_∞) norms are used and B_k is indefinite, the subproblem becomes a non-convex quadratic program for which there is unlikely to be an efficient general-purpose algorithm—in the special case where B_k is positive semi-definite and the ℓ_∞ norm is used, the subproblem is a convex QP. For this reason, the Celis–Dennis–Tapia approach is rarely used in practice.

7.4.3 Filter methods

The last SQP method we shall consider is the most recent. The approach taken is quite radical in that, unlike all of the methods we have considered so far, it makes no use of a merit function to force global convergence. The main objection to merit functions is that they depend, to a large degree, on arbitrary or a-priori unknown parameters. A secondary objection is that they tend to be overly conservative in accepting promising potential iterates. But if we wish to avoid merit functions, we need some other device to encourage convergence. The new idea is to use a "filter"

Let $\theta(x) = \|c(x)\|$ be some norm of the constraint violation at x. A *filter* is a set of pairs $\{(\theta_k, f_k)\}$ of violations and objective values such that no member dominates another, i.e., it does not happen that

$$\theta_i \text{``}<\text{''} \theta_j \text{ and } f_i \text{``}<\text{''} f_j$$

for any pair of filter points $i \neq j$—the "<" here informally means "very slightly smaller than". We illustrate a filter in Figure 7.5. A potential new entry to the "north-east" of any of the existing filter entries would not be permitted, and the forbidden region is the intersection of the solid horizontal and vertical lines emanating to the right and above each filter point. For theoretical reasons (akin to requiring sufficient decrease), we slightly enlarge the forbidden region by putting a small margin around each filter point, and this is illustrated in the figure by the dotted lines.

And now it is clear how to use a filter. Any potential SQP (or other) iterate $x_k + s_k$ will immediately be rejected if it lies in the forbidden filter region accumulated during the previous k iterations. This may be embedded in a trust-region framework, and a typical iteration might be as follows:

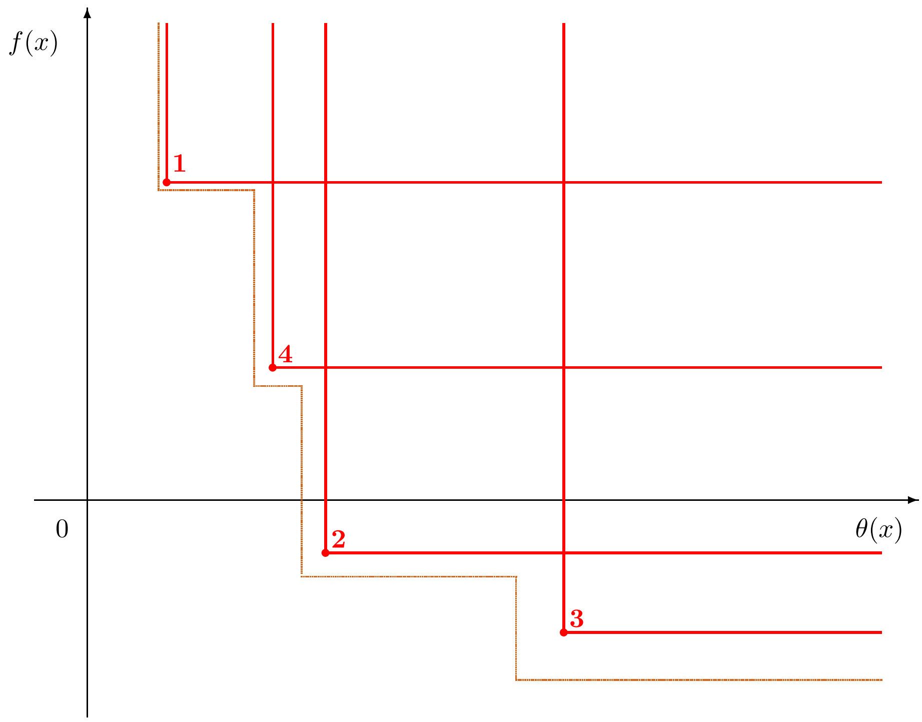

Figure 7.5: A filter with four entries.

If possible find

$$s_k = \underset{s\in\mathbb{R}^n}{\arg\min} \ \langle s, g_k\rangle + \tfrac{1}{2}\langle s, B_k s\rangle \ \text{ subject to } \ A_k s = -c_k \ \text{ and } \ \|s\| \leq \Delta_k,$$

but otherwise, find s_k such that

$$\theta(x_k + s_k) \text{“}<\text{”} \theta_i \ \text{ for all } \ i \leq k.$$

If $x_k + s_k$ is "acceptable" for the filter, set $x_{k+1} = x_k + s_k$
and possibly add $(f((x_k + s_k), \theta(x_k + s_k))$ to the filter,
"prune" the filter, and increase Δ_k.
Otherwise reduce Δ_k and try again.

A few words of explanation are needed. The trust-region and linearized constraints will always be compatible if c_k is small enough so long as they are at $c(x) = 0$. Thus if the trust-region subproblem is incompatible, one remedy is simply to move closer to the constraints. This is known as a *restoration* step. By "pruning" the filter, we mean that a new point may completely dominate one or more existing filter points and, in this case, the dominated entry may be removed without altering the filter. For example, if a new entry were accepted to the “south-west” of point 4 in our figure, point 4 would be pruned.

While the basic filter idea is rather simple, in practice, it is significantly more complicated than this. In particular, there are theoretical reasons why some points that are acceptable to the filter should still be rejected if any decrease in the SQP model of the objective function is far from realized in practice.

CONCLUSIONS

We hope we have conveyed the impression that research into the design, convergence and implementation of algorithms for nonlinear optimization is an exciting and expanding area. We have only been able to outline the developments in the field, and have made no attempt to survey the vast literature that has built up over the last 50 years. Current algorithms for specialized problems like linear and quadratic programming and unconstrained optimization are well capable of solving problems involving millions of unknowns (and constraints), while those for generally constrained optimization routinely solve problems in the tens and, perhaps even, hundreds of thousands of unknowns and constraints. The next big goal is to be able to design algorithms that have some hope of finding global optima for large problems, the current state-of-the-art being for problems with tens or hundreds of unknowns. Clearly closing the gap between local and global optimization has some way to go!

APPENDIX A

SEMINAL BOOKS AND PAPERS

The following books and papers are classics in the field. Although many of them cover topics outside the material we have described, they are all worth reading. This section constitutes a personal view of the most significant papers in the area. It is not meant to be a complete bibliography.

General text books

There are a large number of text books devoted to nonlinear (and even more for linear) programming. Those we find most useful and which emphasize practical methods are

J. Dennis and R. Schnabel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", (republished by) SIAM (Classics in Applied Mathematics 16) (1996),

R. Fletcher, "Practical Methods of Optimization", 2nd edition Wiley (1987), (republished in paperback 2000),

P. Gill, W. Murray and M. Wright, "Practical Optimization", Academic Press (1981), and

J. Nocedal and S. Wright, "Numerical Optimization", Springer Verlag (1999).

The first of these concentrates on unconstrained optimization, while the remainder cover (continuous) optimization in general.

Early quasi-Newton methods

These methods were introduced by

W. Davidon, "Variable metric method for minimization", manuscript (1958), finally published *SIAM J. Optimization* **1** (1991) 1:17,

and championed by

R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization", *Computer J.* (1963) 163:168.

Although the so-called DFP method has been superseded by the more reliable BFGS method, it paved the way for a number of classes of important updates.

More modern quasi-Newton methods

Coincidentally, all of the papers

C. G. Broyden, "The convergence of a class of double-rank minimization algorithms", *J. Inst. Math. Applcs.*, **6** (1970) 76:90,

R. Fletcher, "A new approach to variable metric algorithms", *Computer J.* (1970) **13** (1970) 317:322,

D. Goldfarb, "A family of variable metric methods derived by variational means", *Math. Computation* **24** (1970) 23:26, and

D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization", *Math. Computation* **24** (1970) 647:657

appeared in the same year. The aptly-named BFGS method has stood the test of time well, and is still regarded as possibly the best secant updating formula.

Quasi-Newton methods for large problems

Limited memory methods are secant-updating methods that discard old information so as to reduce the amount of storage required when solving large problems. The methods first appeared in

J. Nocedal, "Updating quasi-Newton matrices with limited storage", *Math. Computation* **35** (1980) 773:782, and

A. Buckley and A. Lenir, "QN-like variable storage conjugate gradients", *Math. Programming* **27** (1983) 155:175.

Secant updating formulae proved to be less useful for large-scale computation, but a successful generalization, applicable to what are known as partially separable functions, was pioneered by

A. Griewank and Ph. Toint, "Partitioned variable metric updates for large structured optimization problems", *Numerische Mathematik* **39** (1982) 119:137, see also 429:448, as well as

A. Griewank and Ph. Toint, "On the unconstrained optimization of partially separable functions", in *Nonlinear Optimization 1981* (Powell, M., ed.) Academic Press (1982)

Conjugate gradient methods for large problems

Generalizations of Conjugate Gradient methods for non-quadratic minimization were originally proposed by

R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients", *Computer J.* (1964) 149:154, and

E. Polak and G. Ribiére, "Note sur la convergence de méthodes de directions conjuguées", *Revue Française d'informatique et de recherche opérationelle* **16** (1969) 35:43.

An alternative is to attempt to solve the (linear) Newton system by a conjugate-gradient like method. Suitable methods for terminating such a procedure while still maintaining fast convergence were proposed by

R. S. Dembo and T. Steihaug, "Truncated-Newton algorithms for large-scale unconstrained optimization", *Math. Programming* **26** (1983) 190:212.

Non-monotone methods

While it is usual to think of requiring that the objective function decreases at every iteration, this is not actually necessary for convergence so long as there is some overall downward trend. The first method along these lines was by

L. Grippo, F. Lampariello and S. Lucidi, "A nonmonotone line search technique for Newton's method", *SIAM J. Num. Anal.*, **23** (1986) 707:716.

Trust-region methods

The earliest methods that might be regarded as trust-region methods are those by

K. Levenberg, "A method for the solution of certain problems in least squares", *Quarterly J. Appl. Maths*, **2** (1944) 164:168, and

D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters" *SIAM J. Appl. Maths*, **11** (1963) 431:441

for the solution of nonlinear least-squares problems, although they are motivated from the perspective of modifying indefinite Hessians rather than restricting the step. Probably the first "modern" interpretation is by

S. Goldfeldt, R. Quandt and H. Trotter, "Maximization by quadratic hill-climbing", *Econometrica*, **34** (1966) 541:551.

Certainly, the earliest proofs of convergence are given by

M. Powell, "A New Algorithm for Unconstrained Optimization", in *Nonlinear Programming*, (Rosen, J., Mangasarian, O., and Ritter, K., eds.) Academic Press (1970),

while a good modern introduction is by

J. Moré, "Recent developments in algorithms and software for trust region methods", in *Mathematical Programming: The State of the Art*, (Bachem, A., Grötschel, M., and Korte, B., eds.) Springer Verlag (1983).

You might want to see our book

A. Conn, N. Gould and Ph. Toint, "Trust-region methods", SIAM (2000)

for a comprehensive history and review of the large variety of articles on trust-region methods.

Trust-region subproblems

Almost all you need to know about solving small-scale trust-region subproblems is contained in the paper

J. Moré and D. Sorensen, "Computing a trust region step", *SIAM J. Sci. Stat. Comp.* **4** (1983) 533:572.

Likewise

T. Steihaug, "The conjugate gradient method and trust regions in large scale optimization", *SIAM J. Num. Anal.* **20** (1983) 626:637

provides the basic truncated conjugate-gradient approach used so successfully for large-scale problems. More recently[1]

N. Gould, S. Lucidi, M. Roma and Ph. Toint, "Solving the trust-region subproblem using the Lanczos method", *SIAM J. Optimization* **9** (1999) 504:525

show how to improve on Steihaug's approach by moving around the trust-region boundary. A particularly nice new paper by

Y. Yuan, "On the truncated conjugate-gradient method", *Math. Programming*, **87** (2000) 561:573

[1] We would hate to claim "seminal" status for one of our own papers!

proves that Steihaug's approximation gives at least 50% of the optimal function decrease when applied to convex problems.

The Symmetric Rank-One quasi-Newton approximation

Since trust-region methods allow non-convex models, perhaps the simplest of all Hessian approximation methods, the Symmetric Rank-One update, is back in fashion. Although it is unclear who first suggested the method,

C. Broyden, "Quasi-Newton methods and their application to function minimization", *Math. Computation* **21** (1967) 577:593

is the earliest reference that we know of. Its revival in fortune is due[2] to

A. Conn, N. Gould and Ph. Toint, "Convergence of quasi-Newton matrices generated by the Symmetric Rank One update" *Math. Programming*, **50** (1991) 177:196 (see also *Math. Comp.* **50** (1988) 399:430), and

R. Byrd, H. Khalfan and R. Schnabel "Analysis of a symmetric rank-one trust region method" *SIAM J. Optimization* **6** (1996) 1025:1039,

and it has now taken its place alongside the BFGS method as the pre-eminent updating formula.

More non-monotone methods

Non-monotone methods have also been proposed in the trust-region case. The basic reference here is the paper by

Ph. Toint, "A non-monotone trust-region algorithm for nonlinear optimization subject to convex constraints", *Math. Programming*, **77** (1997) 69:94.

Barrier function methods

Although they appear to have originated in a pair of unpublished University of Oslo technical reports by K. Frisch in the mid 1950s, (logarithmic) barrier function were popularized by

A. Fiacco and G. McCormick, "The sequential unconstrained minimization technique for nonlinear programming: a primal-dual method", *Management Science* **10** (1964) 360:366; see also *ibid* (1964) 601:617.

A full early history is given in the book

A. Fiacco and G. McCormick, "Nonlinear programming: sequential unconstrained minimization techniques" (1968), republished as *Classics in Applied Mathematics 4*, SIAM (1990).

The worsening conditioning of the Hessian was first highlighted by

F. Lootsma, "Hessian matrices of penalty functions for solving constrained optimization problems", *Philips Research Reports*, **24** (1969) 322:331, and

W. Murray, "Analytical expressions for eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions", *J. Optimization Theory and Applications*, **7** (1971) 189:196,

[2] See previous footnote ...

although recent work by

M. Wright, "Ill-conditioning and computational error in interior methods for nonlinear programming", *SIAM J. Optimization* **9** (1999) 84:111, and

S. Wright, "Effects of finite-precision arithmetic on interior-point methods for nonlinear programming", *SIAM J. Optimization* **12** (2001) 36:78

demonstrates that this "defect" is far from fatal.

Interior-point methods

The interior-point revolution was started by

N. Karmarkar, "A new polynomial-time algorithm for linear programming", *Combinatorica* **4** (1984) 373:395.

It did not take long for

P. Gill, W. Murray, M. Saunders, J. Tomlin and M. Wright, "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method", *Math. Programming*, **36** (1986) 183:209

to realize that this radical "new" approach was actually something that nonlinear programmers had tried (but, most unfortunately, discarded) in the past.

SQP methods

The first SQP method was proposed in the overlooked 1963 Harvard Master's thesis of R. Wilson. The generic linesearch SQP method is that of

B. Pschenichny, "Algorithms for general problems of mathematical programming", *Kibernetica*, **6** (1970) 120:125,

while there is a much larger variety of trust-region SQP methods, principally because of the constraint incompatibility issue.

Merit functions for SQP

The first use of an exact penalty function to globalize the SQP method was by

S. Han, "A globally convergent method for nonlinear programming", *J. Optimization Theory and Applics*, **22** (1977) 297:309, and

M. Powell, "A fast algorithm for nonlinearly constrained optimization calculations", in *Numerical Analysis, Dundee 1977* (G. Watson, ed) Springer Verlag (1978) 144:157.

The fact that such a merit function may prevent full SQP steps was observed N. Maratos in his 1978 U. of London Ph. D. thesis, while methods for combating the Maratos effect were subsequently proposed by

R. Fletcher, "Second-order corrections for non-differentiable optimization", in *Numerical Analysis, Dundee 1981* (G. Watson, ed) Springer Verlag (1982) 85:114, and

R. Chamberlain, M. Powell, C. Lemaréchal, and H. Pedersen, "The watchdog technique for forcing convergence in algorithms for constrained optimization", *Math. Programming Studies*, **16** (1982) 1:17.

An SQP method that avoids the need for a merit function altogether by staying feasible is given by

> J. Bonnans, E. Panier, A. Tits, and J. Zhou, "Avoiding the Maratos effect by means of a nonmonotone linesearch II. Inequality constrained problems—feasible iterates", *SIAM J. Num. Anal.*, **29** (1992) 1187:1202.

Hessian approximations

There is a vast literature on suitable Hessian approximations for use in SQP methods. Rather than point at individual papers, a good place to start is

> P. Boggs and J. Tolle, "Sequential quadratic programming", *Acta Numerica* **4** (1995) 1:51,

but see also our paper

> N. Gould and Ph. Toint, "SQP methods for large-scale nonlinear programming", in *System modelling and optimization, methods, theory and applications* (M. Powell and S. Scholtes, eds.) Kluwer (2000) 149:178.

Trust-region SQP methods

Since the trust-region and the linearized constraints may be incompatible, almost all trust-region SQP methods modify the basic SQP method in some way. The Sℓ_1QP method is due to

> R. Fletcher, "A model algorithm for composite non-differentiable optimization problems", *Math. Programming Studies*, **17** (1982) 67:76.

Methods that relax the constraints include those proposed by

> A. Vardi, "A trust region algorithm for equality constrained minimization: convergence properties and implementation", *SIAM J. Num. Anal.*, **22** (1985) 575:591, and

> M. Celis, J. Dennis and R. Tapia, "A trust region strategy for nonlinear equality constrained optimization", in *Numerical Optimization 1984* (P. Boggs, R. Byrd and R. Schnabel, eds), SIAM (1985) 71:82,

as well as a method that appeared in the 1989 U. of Colorado at Boulder Ph. D. thesis of E. Omojokun, supervised by R. Byrd. The Filter-SQP approach may be found in[3]

> R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function", *Math. Programming*, **91** (2002) 239:269.

Modern methods for nonlinear programming

Many modern methods for nonlinearly constrained optimization tend to be SQP-interior-point hybrids. A good example is due to

> R. Byrd, J. Gilbert and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming", *Math. Programming A* **89** (2000) 149:185,

and forms the basis for the excellent KNITRO package.

[3] Once again, see previous footnote ...

APPENDIX B

OPTIMIZATION RESOURCES ON THE INTERNET

As we are all too keenly aware, the Internet provides a wealth of both information and misinformation on any subject under the sun. Here we suggest some useful resources relating to optimization.

B.1 Answering questions on the web

A good starting point for finding out more about optimization are the two lists of Frequently Asked Questions (FAQs) on optimization. The Linear Programming FAQ,

`www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html` ,

is dedicated to question on linear optimization problems as well as certain aspects of mixed integer linear programming. The Nonlinear Programming FAQ,

`www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html` ,

offers a concise introduction to nonlinear optimization. The NEOS guide,

`www-fp.mcs.anl.gov/otc/Guide` ,

provides an overview of optimization and the solvers available. It contains the optimization tree,

`www-fp.mcs.anl.gov/otc/Guide/OptWeb` ,

a dichotomy of optimization problems. Both sites are maintained by the Optimization Technology Center

`www.ece.nwu.edu/OTC` ,

a loose collaboration between Argonne National Laboratory and Northwestern University in the USA.

Hans Mittelmann of Arizona State University maintains a decision tree for optimization software,

`plato.la.asu.edu/guide.html` ,

and he also provides a useful set of benchmarks for optimization software,

`plato.la.asu.edu/bench.html` .

Harvey Greenberg's Mathematical Programming Glossary,

`www.cudenver.edu/ hgreenbe/glossary/glossary.html`

contains brief definitions of commonly used expressions in optimization and operations research. The usenet newsgroup

`sci.op-research`

is dedicated to answering questions on optimization and operations research. Brian Borchers edits a weekly digest of postings to it. You can receive the digest by sending an email to

`listserv@listserv.okstate.edu`

with the message

`SUBSCRIBE ORCS-L Your Name` .

B.2 Solving optimization problems on the web

B.2.1 The NEOS server

Probably the most important and useful optimization site on the web is the NEOS server[4] at

`www-neos.mcs.anl.gov/neos`

which allows you to solve optimization problems over the internet. NEOS handles several thousand (!) submissions per week. The server provides a wide choice of state-of-the-art optimization software which can be used remotely without the need to install or maintain any software.

The problems should preferably be formulated in a modelling language such as AMPL[5] or GAMS[6] (see Section B.2.3). However, some solvers also accept problem descriptions in other formats such as C or fortran source code or the verbose linear programming MPS format.

There are a number of solvers implementing algorithms for nonlinearly constrained optimization problems. Most are hybrids, and thus capable of handling both equality and inequality constraints. There are at least two solvers implementing augmented Lagrangian ideas (see Part 5).

LANCELOT implements an augmented Lagrangian algorithm. It uses a trust-region to promote global convergence.

MINOS implements a sequential linearly constrained algorithm. Steplength control is heuristic (for want of a suitable merit function), but superlinear convergence is often achieved.

There are at least four interior point solvers (see Part 6).

IPOPT, is a freely available line-search filter primal-dual interior-point method.

KNITRO (with a silent "K"), is a primal-dual interior-point method which uses trust regions.

LOQO is based on an infeasible primal-dual interior-point method. It uses a linesearch and a version of a filter to enforce global convergence.

MOSEK can only be used to solve *convex* large-scale smooth nonlinear optimization problems. It does not work for *nonconvex* problems.

Finally, there are at least three solvers implementing SQP algorithms (see Part 7).

DONLP2 implements a linesearch SQP algorithm with an exact non-differentiable ℓ_1-penalty function as a merit function. It uses dense linear algebra.

FILTER implements a trust-region SQP algorithm which is suitable for solving large nonlinearly constrained problems with small degrees of freedom. It uses a filter (see Part 7.4.3) to promote global convergence.

[4] J. Czyzyk, M. Mesnier and J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998.

[5] R. Fourer, D. Gay and B. Kernighan. *AMPL: A modelling Language for Mathematical Programming.* Boyd & Fraser Publishing Company, Massachusetts, 1993.

[6] A. Brooke, D. Kendrick, A. Meeraus and R. Raman. *GAMS A user's guide.* GAMS Developments Corporation, 1217 Potomac Street, N.W., Washington DC 20007, USA, December 1998.

SNOPT implements a linesearch SQP algorithm which uses an augmented Lagrangian as a merit function. It maintains a positive definite limited memory approximation of the Hessian of the Lagrangian.

There is also a range of other solvers not covered in this article.

CONOPT is a feasible path method based on the generalized reduced gradient algorithm.

PATHNLP finds stationary points for the nonlinear problem by solving the Karush-Kuhn-Tucker conditions (see Theorems 1.7 and 1.9), written as a mixed complementarity problem, using the PATH solver.

Consult the NEOS guide (see Section B.1) for appropriate contacts.

A wide range of other optimization problems can also be solved such as semi-infinite optimization, mixed integer linear and nonlinear optimization, semidefinite optimization, complementarity problems, non-differentiable optimization, and unconstrained and stochastic optimization problems. The fact that the server maintains state-of-the-art optimization software makes is suitable for medium to large scale applications.

Users with their own copy of the modelling systems AMPL or GAMS can even invoke the NEOS solvers out of their local AMPL or GAMS session using KESTREL,

`www-neos.mcs.anl.gov/neos/kestrel.html` .

This is very convenient as it makes it possible to post- or pre-process the models using a local copy of the modelling tool.

B.2.2 Other online solvers

The system www-Nimbus, from

`nimbus.mit.jyu.fi` ,

is designed to solve (small) multi-objective optimization problems. It consists of a sequence of menus to input the multi-objective problem as well as some facilities for displaying the solution. It requires the user to interactively guide the optimization and requires some familiarity with multi-objective terminology. An online tutorial guides the user through the process. Certain topology optimization problems can be solved at

`www.topopt.dtu.dk`

The input is via a GUI and the solution is also display graphically. The system Baron,

`archimedes.scs.uiuc.edu/baron/availability.html` ,

allows the solution of small global optimization problems online.

B.2.3 Useful sites for modelling problems prior to online solution

AMPL (A Mathematical Programming Language)

`www.ampl.com`

is a modelling language for optimization problems. The site lists extensions to the book, allows the solution of example models and contains a list of available solvers. Further AMPL models can be found at the following sites:

NLP models by Bob Vanderbei:

`www.sor.princeton.edu/~rvdb/ampl/nlmodels` .

MINLP and MPEC models by Sven Leyffer:

`www.maths.dundee.ac.uk/~sleyffer/MacMINLP` and
`www.maths.dundee.ac.uk/~sleyffer/MacMPEC` .

The COPS collection of Jorge Moré:

`www-unix.mcs.anl.gov/~more/cops` .

These sites are especially useful to help with your own modelling exercises.

GAMS (the General Algebraic Modelling System)

`www.gams.com`

is another modelling language. The site contains documentation on GAMS and some example models. More GAMS models can be found on the GAMS-world pages. These are sites, dedicated to important modelling areas, see

`www.gamsworld.org` .

It also offers a translation service from one modelling language to another.

Recently, optimization solvers have also been interfaced to `matlab` at

`tomlab.biz/` .

B.2.4 Free optimization software

An extension of MPS to nonlinear optimization, SIF (standard input format), can be used to model optimization problems. The reference document can be found at

`www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html` .

A collection of optimization problems in SIF is available at CUTEr can be found via

`www.cse.clrc.ac.uk/Activity/CUTEr` .

Two solvers, LANCELOT

`www.cse.clrc.ac.uk/Activity/LANCELOT`

and GALAHAD

`www.cse.clrc.ac.uk/Activity/GALAHAD`

are available freely for non-commercial users.

AMPL and some solvers are also available freely in limited size student versions, which allow the solution of problems of up to 300 variables and constraints, see

`netlib.bell-labs.com/netlib/ampl/student/` .

B.3 Optimization reports on the web

Optimization online,

`www.optimization-online.org` ,

is an e-print site for papers on optimization. It is sponsored by the Mathematical Programming Society. It allows you to search for preprints on certain subjects. A monthly digest summarizes all monthly submissions.

The two main optimization journals, Mathematical Programming and SIAM Journal on Optimization maintain free sites with access to titles and abstracts, see

`link.springer.de/link/service/journals/10107/`,

and

`www.siam.org/journals/siopt/siopt.htm` .

APPENDIX C

SKETCHES OF PROOFS

As promised, we now give sketches of proofs of almost all the results we have stated. For the few results that are simply too complicated to summarize, we indicate where the reader might find the complete proof.

Theorems 1.1—1.3 can be found in any good book on analysis. Theorems 1.1 and 1.2 follow directly by considering the remainders of truncated Taylor expansions of the univariate function $f(x + \alpha s)$ with $\alpha \in [0, 1]$, while Theorem 1.3 uses the Newton formula

$$F(x+s) = F(x) + \int_0^1 \nabla_x F(x + \alpha s) s d\alpha.$$

Proof of Farkas' lemma

The result is trivial if $\mathcal{C} = 0$. So otherwise, suppose that $g \in \mathcal{C}$ and that $\langle s, a_i \rangle \geq 0$ for $i \in \mathcal{A}$. Then

$$\langle s, g \rangle = \sum_{i \in \mathcal{A}} y_i \langle s, a_i \rangle \geq 0.$$

Hence $\mathcal{S}$ is empty, since $\langle s, g \rangle$ is non-negative.

Conversely, suppose that $g \notin \mathcal{C}$, and consider

$$\min_{c \in \mathcal{C}} \|g - c\|_2 = \min_{c \in \bar{\mathcal{C}}} \|g - c\|_2,$$

where

$$\bar{\mathcal{C}} = \mathcal{C} \bigcap \{c \mid \|g - c\|_2 \leq \|g - \bar{c}\|_2\}$$

and $\bar{c}$ is any point in $\mathcal{C}$. Since $\mathcal{C}$ is closed, and $\{c \mid \|g - c\|_2 \leq \|g - \bar{c}\|_2\}$ is compact, $\bar{C}$ is non-empty and compact, and it follows from Weierstrass' Theorem (namely, that the minimizer of continuous function within a compact set is achieved) that

$$c_* = \arg\min_{c \in \mathcal{C}} \|g - c\|_2$$

exists. As $\mathcal{C}$ is convex with 0, $c_* \in \mathcal{C}$, $\alpha c_* \in \mathcal{C}$ for all $\alpha \geq 0$, and hence $\phi(\alpha) = \|g - \alpha c_*\|_2^2$ is minimized at $\alpha = 1$. Hence $\phi'(1) = 0$ and thus

$$\langle c_*, c_* - g \rangle = 0. \tag{C.1}$$

By convexity, if $c \in \mathcal{C}$, so is $c_* + \theta(c - c_*)$ for all $\theta \in [0, 1]$, and hence by optimality of c_*

$$\|g - c_*\|_2^2 \leq \|g - c_* + \theta(c_* - c)\|_2^2.$$

Expanding and taking the limit as θ approaches zero, we deduce that

$$0 \leq \langle g - c_*, c_* - c \rangle = \langle c_* - g, c \rangle$$

using (C.1). Thus, defining $s = c_* - g$, $\langle s, c \rangle \geq 0$ for all $c \in \mathcal{C}$, and in particular $\langle s, a_i \rangle \geq 0$ for all $i \in \mathcal{A}$. But as $s \neq 0$, as $c_* \in \mathcal{C}$ and $g \notin \mathcal{C}$, and $\langle s, g \rangle = -\langle s, s \rangle < 0$, using (C.1), we have exhibited the separating hyperplane $\langle s, v \rangle = 0$ as required when $g \notin \mathcal{C}$.

Proof of Theorem 1.4

Suppose otherwise, that $g(x_*) \neq 0$. A Taylor expansion in the direction $-g(x_*)$ gives

$$f(x_* - \alpha g(x_*)) = f(x_*) - \alpha \|g(x_*)\|^2 + O(\alpha^2).$$

For sufficiently small α, $\frac{1}{2}\alpha\|g(x_*)\|^2 \geq O(\alpha^2)$, and thus

$$f(x_* - \alpha g(x_*)) \leq f(x_*) - \tfrac{1}{2}\alpha\|g(x_*)\|^2 < f(x_*).$$

This contradicts the hypothesis that x_* is a local minimizer.

Proof of Theorem 1.5

Again, suppose otherwise that $\langle s, H(x_*)s\rangle < 0$. A Taylor expansion in the direction s gives

$$f(x_* + \alpha s) = f(x_*) + \tfrac{1}{2}\alpha^2\langle s, H(x_*)s\rangle + O(\alpha^3),$$

since $g(x_*) = 0$. For sufficiently small α, $-\frac{1}{4}\alpha^2\langle s, H(x_*)s\rangle \geq O(\alpha^3)$, and thus

$$f(x_* + \alpha s) \leq f(x_*) + \tfrac{1}{4}\alpha^2\langle s, H(x_*)s\rangle < f(x_*).$$

Once again, this contradicts the hypothesis that x_* is a local minimizer.

Proof of Theorem 1.6

By continuity $H(x)$ is positive definite for all x in a open ball $\mathcal{N}$ around x_*. The generalized mean-value theorem then says that if $x_* + s \in \mathcal{N}$, there is a value z between the points x_* and $x_* + s$ for which

$$f(x_* + s) = f(x_*) + \langle s, g(x_*)\rangle + \tfrac{1}{2}\langle s, H(z)s\rangle = f(x_*) + \tfrac{1}{2}\langle s, H(z)s\rangle > f(x_*)$$

for all nonzero s, and thus x_* is an isolated local minimizer.

Proof of Theorem 1.7

We consider feasible perturbations about x_*. Consider a vector valued C^2 (C^3 for Theorem 1.8) function $x(\alpha)$ of the scalar α for which $x(0) = x_*$ and $c(x(\alpha)) = 0$. (The constraint qualification is that all such feasible perturbations are of this form). We may then write

$$x(\alpha) = x_* + \alpha s + \tfrac{1}{2}\alpha^2 p + O(\alpha^3) \tag{C.2}$$

and we require that

$$\begin{aligned}
0 &= c_i(x(\alpha)) = c_i(x_* + \alpha s + \tfrac{1}{2}\alpha^2 p + O(\alpha^3)) \\
&= c_i(x_*) + \langle a_i(x_*), \alpha s + \tfrac{1}{2}\alpha^2 p\rangle + \tfrac{1}{2}\alpha^2\langle s, H_i(x_*)s\rangle + O(\alpha^3) \\
&= \alpha\langle a_i(x_*), s\rangle + \tfrac{1}{2}\alpha^2\left(\langle a_i(x_*), p\rangle + \langle s, H_i(x_*)s\rangle\right) + O(\alpha^3)
\end{aligned}$$

using Taylor's theorem. Matching similar asymptotic terms, this implies that for such a feasible perturbation

$$A(x_*)s = 0 \tag{C.3}$$

and

$$\langle a_i(x_*), p\rangle + \langle s, H_i(x_*)s\rangle = 0 \tag{C.4}$$

for all $i = 1, \ldots, m$. Now consider the objective function

$$\begin{aligned} f(x(\alpha)) &= f(x_* + \alpha s + \tfrac{1}{2}\alpha^2 p + O(\alpha^3)) \\ &= f(x_*) + \langle g(x_*), \alpha s + \tfrac{1}{2}\alpha^2 p\rangle + \tfrac{1}{2}\alpha^2\langle s, H(x_*)s\rangle + O(\alpha^3) \\ &= f(x_*) + \alpha\langle g(x_*), s\rangle + \tfrac{1}{2}\alpha^2\left(\langle g(x_*), p\rangle + \langle s, H(x_*)s\rangle\right) + O(\alpha^3). \end{aligned} \tag{C.5}$$

This function is unconstrained along $x(\alpha)$, so we may deduce, as in Theorem 1.4, that

$$\langle g(x_*), s\rangle = 0 \text{ for all } s \text{ such that } A(x_*)s = 0. \tag{C.6}$$

If we let S be a basis for the null-space of $A(x_*)$, we may write

$$g(x_*) = A^T(x_*)y_* + Sz_* \tag{C.7}$$

for some y_* and z_*. Since, by definition, $A(x_*)S = 0$, and as it then follows from (C.6) that $g^T(x_*)S = 0$, we have that

$$0 = S^T g(x_*) = S^T A^T(x_*)y_* + S^T S z_* = S^T S z_*.$$

Hence $S^T S z_* = 0$ and thus $z_* = 0$ since S is of full rank. Thus (C.7) gives

$$g(x_*) - A^T(x_*)y_* = 0. \tag{C.8}$$

Proof of Theorem 1.8

We have shown that

$$f(x(\alpha)) = f(x_*) + \tfrac{1}{2}\alpha^2\left(\langle p, g(x_*)\rangle + \langle s, H(x_*)s\rangle\right) + O(\alpha^3) \tag{C.9}$$

for all s satisfying $A(x_*)s = 0$, and that (C.8) holds. Hence, necessarily,

$$\langle p, g(x_*)\rangle + \langle s, H(x_*)s\rangle \geq 0 \tag{C.10}$$

for all s and p satisfying (C.3) and (C.4). But (C.8) and (C.4) combine to give

$$\langle p, g(x_*)\rangle = \sum_{i=1}^m (y_*)_i \langle p, a_i(x_*)\rangle = -\sum_{i=1}^m (y_*)_i \langle s, H_i(x_*)s\rangle$$

and thus (C.10) is equivalent to

$$\left\langle s, \left(H(x_*) - \sum_{i=1}^m (y_*)_i H_i(x_*)\right) s\right\rangle \equiv \langle s, H(x_*, y_*)s\rangle \geq 0$$

for all s satisfying (C.3).

Proof of Theorem 1.9

As in the proof of Theorem 1.7, we consider feasible perturbations about x_*. Since any constraint that is inactive at x_* (i.e., $c_i(x_*) > 0$) will remain inactive for small perturbations, we need only consider perturbations that are constrained by the constraints active at x_*, (i.e., $c_i(x_*) = 0$). Let $\mathcal{A}$ denote the indices of the active constraints. We then consider a vector valued C^2 (C^3 for Theorem 1.10) function $x(\alpha)$ of the scalar α for which $x(0) = x_*$ and $c_i(x(\alpha)) \geq 0$ for $i \in \mathcal{A}$. In this case, assuming that $x(\alpha)$ may be expressed as (C.2), we require that

$$\begin{aligned} 0 \;\leq\;& c_i(x(\alpha)) = c(x_* + \alpha s + \tfrac{1}{2}\alpha^2 p + O(\alpha^3)) \\ =\;& c_i(x_*) + \langle a_i(x_*), \alpha s + \tfrac{1}{2}\alpha^2 p\rangle + \tfrac{1}{2}\alpha^2 \langle s, H_i(x_*)s\rangle + O(\alpha^3) \\ =\;& \alpha\langle a_i(x_*), s\rangle + \tfrac{1}{2}\alpha^2\left(\langle a_i(x_*), p\rangle + \langle s, H_i(x_*)s\rangle\right) + O(\alpha^3) \end{aligned}$$

for all $i \in \mathcal{A}$. Thus

$$\langle s, a_i(x_*)\rangle \geq 0 \tag{C.11}$$

and

$$\langle p, a_i(x_*)\rangle + \langle s, H_i(x_*)s\rangle \geq 0 \;\text{ when }\; \langle s, a_i(x_*)\rangle = 0 \tag{C.12}$$

for all $i \in \mathcal{A}$. The expansion of $f(x(\alpha))$ (C.5) then implies that x_* can only be a local minimizer if

$$\mathcal{S} = \{s \mid \langle s, g(x_*)\rangle < 0 \;\text{ and }\; \langle s, a_i(x_*)\rangle \geq 0 \;\text{ for }\; i \in \mathcal{A}\} = \emptyset.$$

But then the result follows directly from Farkas' lemma.

Proof of Theorem 1.10

The expansion (C.5) for the change in the objective function will be dominated by the first-order term $\alpha\langle s, g(x_*)\rangle$ for feasible perturbations unless $\langle s, g(x_*)\rangle = 0$, in which case the expansion (C.9) is relevant. Thus we must have that (C.10) holds for all feasible s for which $\langle s, g(x_*)\rangle = 0$. The latter requirement gives that

$$0 = \langle s, g(x_*)\rangle = \sum_{i\in\mathcal{A}} y_i \langle s, a_i(x_*)\rangle,$$

and hence that either $y_i = 0$ or $\langle s, a_i(x_*)\rangle = 0$ (or both).

We now focus on the *subset* of all feasible arcs that ensure $c_i(x(\alpha)) = 0$ if $y_i > 0$ and $c_i(x(\alpha)) \geq 0$ if $y_i = 0$ for $i \in \mathcal{A}$. For those constraints for which $c_i(x(\alpha) = 0$, we have that (C.3) and (C.4) hold, and thus for such perturbations $s \in \mathcal{N}_+$. In this case

$$\langle p, g(x_*)\rangle = \sum_{i\in\mathcal{A}} y_i\langle p, a_i(x_*)\rangle = \sum_{\substack{i\in\mathcal{A}\\ y_i>0}} y_i\langle p, a_i(x_*)\rangle = -\sum_{\substack{i\in\mathcal{A}\\ y_i>0}} y_i\langle s, H_i(x_*)s\rangle = -\sum_{i\in\mathcal{A}} y_i\langle s, H_i(x_*)s\rangle$$

This combines with (C.10) to give that

$$\langle s, H(x_*, y_*)s\rangle \equiv \left\langle s, \left(H(x_*) - \sum_{i=1}^m (y_*)_i H_i(x_*)\right) s\right\rangle = \langle p, g(x_*)\rangle + \langle s, H(x_*)s\rangle \geq 0.$$

for all $s \in \mathcal{N}_+$, which is the required result.

Proof of Theorem 1.11

Consider any feasible arc $x(\alpha)$. We have seen that (C.11) and (C.12) hold, and that first-order feasible perturbations are characterized by $\mathcal{N}_+$. It then follows from (C.12) that

$$\langle p, g(x_*)\rangle = \sum_{i\in\mathcal{A}} y_i\langle p, a_i(x_*)\rangle = \sum_{\substack{i\in\mathcal{A}\\ \langle s,a_i(x_*)\rangle=0}} y_i\langle p, a_i(x_*)\rangle \geq -\sum_{\substack{i\in\mathcal{A}\\ \langle s,a_i(x_*)\rangle=0}} y_i\langle s, H_i(x_*)s\rangle = -\sum_{i\in\mathcal{A}} y_i\langle s, H_i(x_*)s\rangle,$$

and hence by assumption that

$$\langle p, g(x_*)\rangle + \langle s, H(x_*)s\rangle \geq \left\langle s, \left(H(x_*) - \sum_{i=1}^{m}(y_*)_i H_i(x_*)\right) s \right\rangle \equiv \langle s, H(x_*, y_*)s\rangle > 0$$

for all $s \in \mathcal{N}_+$. But this then combines with (C.5) and (C.11) to show that $f(x(\alpha)) > f(x_*)$ for all sufficiently small α.

Proof of Theorem 2.1

From Taylor's theorem (Theorem 1.1), and using the bound

$$\alpha \leq \frac{2(\beta-1)\langle p, g(x)\rangle}{\gamma(x)\|p\|_2^2},$$

we have that

$$\begin{aligned} f(x+\alpha p) &\leq f(x) + \alpha\langle p, g(x)\rangle + \tfrac{1}{2}\gamma(x)\alpha^2\|p\|^2 \\ &\leq f(x) + \alpha\langle p, g(x)\rangle + \alpha(\beta-1)\langle p, g(x)\rangle \\ &= f(x) + \alpha\beta\langle p, g(x)\rangle. \end{aligned}$$

Proof of Corollary 2.2

Theorem 2.1 shows that the linesearch will terminate as soon as $\alpha^{(l)} \leq \alpha_{\max}$. There are two cases to consider. Firstly, it may be that α_{init} satisfies the Armijo condition, in which case $\alpha_k = \alpha_{\text{init}}$. If not, there must be a last linesearch iteration, say the lth, for which $\alpha^{(l)} > \alpha_{\max}$ (if the linesearch has not already terminated). Then $\alpha_k \geq \alpha^{(l+1)} = \tau\alpha^{(l)} > \tau\alpha_{\max}$. Combining these two cases gives the required result.

Proof of Theorem 2.3

We shall suppose that $g_k \neq 0$ for all k and that

$$\lim_{k\to\infty} f_k > -\infty.$$

From the Armijo condition, we have that

$$f_{k+1} - f_k \leq \alpha_k\beta\langle p_k, g_k\rangle$$

for all k, and hence summing over the first j iterations

$$f_{j+1} - f_0 \leq \sum_{k=0}^{j} \alpha_k \beta \langle p_k, g_k \rangle.$$

Since the left-hand side of this inequality is, by assumption, bounded below, so is the sum on right-hand-side. As this sum is composed of negative terms, we deduce that

$$\lim_{k \to \infty} \alpha_k \langle p_k, g_k \rangle = 0.$$

Now define the two sets

$$\mathcal{K}_1 = \left\{ k \mid \alpha_{\text{init}} > \frac{2\tau(\beta - 1)\langle p_k, g_k \rangle}{\gamma \|p_k\|_2^2} \right\}$$

and

$$\mathcal{K}_2 = \left\{ k \mid \alpha_{\text{init}} \leq \frac{2\tau(\beta - 1)\langle p_k, g_k \rangle}{\gamma \|p_k\|_2^2} \right\},$$

where γ is the assumed uniform Lipschitz constant. For $k \in \mathcal{K}_1$,

$$\alpha_k \geq \frac{2\tau(\beta - 1)\langle p_k, g_k \rangle}{\gamma \|p_k\|_2^2}$$

in which case

$$\alpha_k \langle p_k, g_k \rangle \leq \frac{2\tau(\beta - 1)}{\gamma} \left(\frac{\langle p_k, g_k \rangle}{\|p_k\|} \right)^2 < 0.$$

Thus

$$\lim_{k \in \mathcal{K}_1 \to \infty} \frac{|\langle p_k, g_k \rangle|}{\|p_k\|_2} = 0. \tag{C.13}$$

For $k \in \mathcal{K}_2$,

$$\alpha_k \geq \alpha_{\text{init}}$$

in which case

$$\lim_{k \in \mathcal{K}_2 \to \infty} |\langle p_k, g_k \rangle| = 0. \tag{C.14}$$

Combining (C.13) and (C.14) gives the required result.

Proof of Theorem 2.4

Follows immediately from Theorem 2.3, since for $p_k = -g_k$,

$$\min\left(|\langle p_k, g_k \rangle|, |\langle p_k, g_k \rangle| / \|p_k\|_2\right) = \|g_k\|_2 \min\left(1, \|g_k\|_2\right)$$

and thus

$$\lim_{k \to \infty} \min\left(|\langle p_k, g_k \rangle|, |\langle p_k, g_k \rangle| / \|p_k\|_2\right) = 0$$

implies that $\lim_{k \to \infty} g_k = 0$.

Proof of Theorem 2.5

Let $\lambda^{\min}(B_k)$ and $\lambda^{\max}(B_k)$ be the smallest and largest eigenvalues of B_k. By assumption, there are bounds $\lambda^{\min} > 0$ and $\lambda^{\max}$ such that

$$\lambda^{\min} \leq \lambda^{\min}(B_k) \leq \frac{\langle s, B_k s\rangle}{\|s\|^2} \leq \lambda^{\max}(B_k) \leq \lambda^{\max}$$

for any nonzero vector s. Thus

$$|\langle p_k, g_k\rangle| = |\langle g_k, B_k^{-1} g_k\rangle| \geq \lambda_{\min}(B_k^{-1})\|g_k\|_2^2 = \frac{1}{\lambda_{\max}(B_k)}\|g_k\|_2^2 \geq \lambda_{\max}^{-1}\|g_k\|_2^2.$$

In addition

$$\|p_k\|_2^2 = \langle g_k, B_k^{-2} g_k\rangle \leq \lambda_{\max}(B_k^{-2})\|g_k\|_2^2 = \frac{1}{\lambda_{\min}(B_k^2)}\|g_k\|_2^2 \leq \lambda_{\min}^{-2}\|g_k\|_2^2,$$

and hence

$$\|p_k\|_2 \leq \lambda_{\min}^{-1}\|g_k\|_2,$$

which leads to

$$\frac{|\langle p_k, g_k\rangle|}{\|p_k\|_2} \geq \frac{\lambda_{\min}}{\lambda_{\max}}\|g_k\|_2.$$

Thus

$$\min\left(|\langle p_k, g_k\rangle|, |\langle p_k, g_k\rangle|/\|p_k\|_2\right) \geq \lambda_{\max}^{-1}\|g_k\|_2 \min\left(\|g_k\|_2, \lambda_{\min}\right).$$

and hence

$$\lim_{k\to\infty} \min\left(|\langle p_k, g_k\rangle|, |\langle p_k, g_k\rangle|/\|p_k\|_2\right) = 0$$

implies, as before, that $\lim_{k\to\infty} g_k = 0$.

Proof of Theorem 2.6

Consider the sequence of iterates x_k, $k \in \mathcal{K}$, whose limit is x_*. By continuity, H_k is positive definite for all such k sufficiently large. In particular, we have that there is a $k_0 \geq 0$ such that

$$\langle p_k, H_k p_k\rangle \geq \tfrac{1}{2}\lambda_{\min}(H_*)\|p_k\|_2^2$$

for all $k \in \mathcal{K} \geq k_0$, where $\lambda_{\min}(H_*)$ is the smallest eigenvalue of $H(x_*)$. We may then deduce that

$$|\langle p_k, g_k\rangle| = -\langle p_k, g_k\rangle = \langle p_k, H_k p_k\rangle \geq \tfrac{1}{2}\lambda_{\min}(H_*)\|p_k\|_2^2. \tag{C.15}$$

for all such k, and also that

$$\lim_{k\in\mathcal{K}\to\infty} p_k = 0$$

since Theorem 2.5 implies that at least one of the left-hand sides of (C.15) and

$$\frac{|\langle p_k, g_k\rangle|}{\|p_k\|_2} = -\frac{\langle p_k, g_k\rangle}{\|p_k\|_2} \geq \tfrac{1}{2}\lambda_{\min}(H_*)\|p_k\|_2$$

converges to zero for all such k.

From Taylor's theorem, there is a z_k between x_k and $x_k + p_k$ such that

$$f(x_k + p_k) = f_k + \langle p_k, g_k \rangle + \tfrac{1}{2}\langle p_k, H(z_k)p_k \rangle.$$

Thus, the Lipschitz continuity of H gives that

$$\begin{aligned} f(x_k + p_k) - f_k - \tfrac{1}{2}\langle p_k, g_k \rangle &= \tfrac{1}{2}(\langle p_k, g_k \rangle + \langle p_k, H(z_k)p_k \rangle) \\ &= \tfrac{1}{2}(\langle p_k, g_k \rangle + \langle p_k, H_k p_k \rangle) + \tfrac{1}{2}\langle p_k, (H(z_k) - H_k)p_k \rangle \\ &\le \tfrac{1}{2}\gamma \|z_k - x_k\|_2 \|p_k\|_2^2 \le \tfrac{1}{2}\gamma \|p_k\|_2^3 \end{aligned} \tag{C.16}$$

since $H_k p_k + g_k = 0$. Now pick k sufficiently large so that

$$\gamma \|p_k\|_2 \le \lambda_{\min}(H_*)(1 - 2\beta).$$

In this case, (C.15) and (C.16) give that

$$f(x_k + p_k) - f_k \le \tfrac{1}{2}\langle p_k, g_k \rangle + \tfrac{1}{2}\lambda_{\min}(H_*)(1 - 2\beta)\|p_k\|_2^2 \le \tfrac{1}{2}(1 - (1 - 2\beta))\langle p_k, g_k \rangle = \beta \langle p_k, g_k \rangle,$$

and thus that a unit stepsize satisfies the Armijo condition for all sufficiently large $k \in \mathcal{K}$.

Now note that $\|H_k^{-1}\|_2 \le 2/\lambda_{\min}(H_*)$ for all sufficiently large $k \in \mathcal{K}$. The iteration gives

$$x_{k+1} - x_* = x_k - x_* - H_k^{-1} g_k = x_k - x_* - H_k^{-1}(g_k - g(x_*)) = H_k^{-1}(g(x_*) - g_k - H_k(x_* - x_k)).$$

But Theorem 1.3 gives that

$$\|g(x_*) - g_k - H_k(x_* - x_k)\|_2 \le \gamma \|x_* - x_k\|_2^2.$$

Hence

$$\|x_{k+1} - x_*\|_2 \le \gamma \|H_k^{-1}\|_2 \|x_* - x_k\|_2^2$$

which is (iii) when $\kappa = 2\gamma/\lambda_{\min}(H_*)$ for $k \in \mathcal{K}$. Result (ii) follows since once an iterate becomes sufficiently close to x_* for sufficiently large $k \in \mathcal{K}$, this implies $k + 1 \in \mathcal{K}$, and hence $\mathcal{K} = \mathbb{N}$. Thus (i) and (iii) are true for all k sufficiently large.

Conjugate-gradient methods (Part 2.5.4)

All of the results given in Part 2.5.4 are easy to verify. For completeness, we include formal statements and sketches of proofs here.

Lemma C.2.1. Suppose that $D^i = (d^0 : \cdots : d^{i-1})$, $\mathcal{D}^i = \{p \mid p = D^i p_d \text{ for some } p_d \in \mathbb{R}^i\}$ and $p^i = \arg\min_{p \in \mathcal{D}^i} q(p) = \langle p, g \rangle + \tfrac{1}{2}\langle p, Bp \rangle$. Then

$$\langle d^j, g^i \rangle = 0 \text{ for } j = 0, \ldots, i - 1, \tag{C.17}$$

where

$$g^i = Bp^i + g. \tag{C.18}$$

Proof: We require $p^i = D^i p_d^i$, where $p_d^i = \underset{p_d \in \mathbb{R}^i}{\arg\min}\; q(D^i p_d)$. Since $q(D^i p_d) = \langle p_d, D^{i\,T} g\rangle + \frac{1}{2}\langle p_d, D^{i\,T} B D^i p_d\rangle$, stationarity of the gradient of $q(D^i p_d)$ yields

$$0 = D^{i\,T} B D^i p_d^i + D^{i\,T} g = D^{i\,T}(BD^i p_d^i + g) = D^{i\,T}(Bp^i + g) = D^{i\,T} g^i,$$

which is (C.17) in vector form.

Lemma C.2.2. Under the assumptions of Lemma C.2.1,

$$p^i = p^{i-1} - \langle d^{i-1}, g^{i-1}\rangle D^i (D^{i\,T} B D^i)^{-1} e_i. \tag{C.19}$$

Proof. Clearly $p^{i-1} \in \mathcal{D}^{i-1} \subset \mathcal{D}^i$, and thus we require $p^i = p^{i-1} + D^i p_d^i$, where $p_d^i = \underset{p_d \in \mathbb{R}^i}{\arg\min}\; q(p^{i-1} + D^i p_d)$. But

$$\begin{aligned} q(p^{i-1} + D^i p_d) &= q(p^{i-1}) + \langle p_d, D^{i\,T}(g + Bp^{i-1})\rangle + \tfrac{1}{2}\langle p_d, D^{i\,T} B D^i p_d\rangle \\ &= q(p^{i-1}) + \langle p_d, D^{i\,T} g^{i-1}\rangle + \tfrac{1}{2}\langle p_d, D^{i\,T} B D^i p_d\rangle \\ &= q(p^{i-1}) + \langle d^{i-1}, g^{i-1}\rangle \langle p_d, e_i\rangle + \tfrac{1}{2}\langle p_d, D^{i\,T} B D^i p_d\rangle. \end{aligned}$$

Stationarity of the gradient of $q(p^{i-1} + D^i p_d)$ then gives the required result $p_d^i = -d^{i-1\,T} g^{i-1} (D^{i\,T} B D^i)^{-1} e_i$.

Lemma C.2.3. Suppose, in addition to the assumptions of Lemma C.2.1, that

$$\langle d^i, Bd^j\rangle = 0 \;\text{ for }\; i \neq j. \tag{C.20}$$

Then

$$p^i = p^{i-1} + \alpha^{i-1} d^{i-1}, \tag{C.21}$$

where

$$\alpha^{i-1} = -\frac{\langle d^{i-1}, g^{i-1}\rangle}{\langle d^{i-1}, Bd^{i-1}\rangle}. \tag{C.22}$$

Proof. The B-conjugacy (C.20) of the $\{d^j\}$ implies that $D^{i\,T} B D^i$ is a diagonal matrix with diagonal entries $\langle d^j, Bd^j\rangle$ for $j = 0, \ldots i-1$. Thus $(D^{i\,T} B D^i)^{-1} e_i = e_i / \langle d^{i-1}, Bd^{i-1}\rangle$. The result then follows directly from (C.19).

Lemma C.2.4. (Orthogonal gradients) Suppose, in addition to the assumptions of Lemma C.2.3, that

$$d^i = -g^i + \sum_{j=0}^{i-1} \beta^{ij} d^j. \tag{C.23}$$

Then

$$\langle g^i, g^j\rangle = 0 \;\text{ for all }\; i \neq j. \tag{C.24}$$

Proof. It follows directly by induction from (C.23) that $\text{span}\{g^i\} = \text{span}\{d^i\}$. Thus $g^j = \sum_{k=0}^{j} \gamma^{j,k} d^k$ for some $\gamma^{j,k}$, and hence from (C.17) that $\langle g^i, g^j\rangle = \sum_{k=0}^{j} \gamma^{j,k} \langle g^i, d^k\rangle = 0$ when $j < i$.

Lemma C.2.5. Under the assumptions of Lemma C.2.4,

$$\langle d^i, g^i\rangle = \langle d^i, g\rangle \tag{C.25}$$

Proof. It follows from (C.21) that $p^i = \sum_{j=0}^{i-1} \alpha_j d^j$. Thus (C.18) and (C.20) together give

$$\langle d^i, g^i \rangle = \langle d^i, g + Bp^i \rangle = \langle d^i, g + \sum_{j=0}^{i-1} \alpha_j \langle d^i, Bd^j \rangle = \langle d^i, g \rangle \rangle$$

Lemma C.2.6. Under the assumptions of Lemma C.2.4,

$$\langle d^i, g^i \rangle = -\|g^i\|_2^2. \tag{C.26}$$

Proof. It follows directly from (C.23) that $\langle g^i, d^i \rangle = -\langle g^i, g^i \rangle + \sum_{j=0}^{i-1} \beta^{ij} \langle g^i, d^j \rangle$. The result then follows immediately from (C.17).

Corollary C.2.7. Under the assumptions of Lemma C.2.4,

$$\alpha^i = \frac{\|g^i\|_2^2}{\langle d^i, Bd^i \rangle}. \tag{C.27}$$

Proof. This is immediate from (C.24) and (C.26).

Lemma C.2.8. Under the assumptions of Lemma C.2.4,

$$\langle g^i, Bd^j \rangle = \begin{cases} 0 & \text{if } j < i-1 \\ \dfrac{\|g^i\|_2^2}{\alpha^{i-1}} & \text{if } j = i-1 \end{cases} \tag{C.28}$$

Proof. It follows directly from (C.18) and (C.21) that $g^{j+1} = g^j + \alpha^j Bd^j$, and thus that $\langle g^i, g^{j+1} \rangle = \langle g^i, g^j \rangle + \alpha^j \langle g^i, Bd^j \rangle$. But then (C.24) implies that $\langle g^i, Bd^j \rangle = 0$ if $j < i-1$, while, if $j = i-1$, $\langle g^i, g^i \rangle = \langle g^i, g^{i-1} \rangle + \alpha^{i-1} \langle g^i, Bd^{i-1} \rangle$ from which we deduce (C.28) once again using (C.24).

Lemma C.2.9. Under the assumptions of Lemma C.2.4,

$$\beta^{ij} = \begin{cases} 0 & \text{if } j < i-1 \\ \beta^i \equiv \dfrac{\|g_i\|_2^2}{\|g_{i-1}\|_2^2} & \text{if } j = i-1 \end{cases}$$

Proof. The required B-conjugacy (C.20) and (C.23) give that

$$0 = \langle d^j, Bd^i \rangle = -\langle d^j, Bg^i \rangle + \sum_{k=0}^{i-1} \beta^{ik} \langle d^j, Bd^k \rangle = -\langle d^j, Bg^i \rangle + \beta^{ij} \langle d^j, Bd^j \rangle$$

and thus that

$$\beta^{ij} = \langle d^j, Bg^i \rangle / \langle d^j, Bd^j \rangle.$$

The result follows immediately from (C.28) for $j < i-1$. For $j = i-1$, again using (C.27) and now also (C.27), as required we have

$$\beta^{i\,i-1} = \frac{\langle d^{i-1\}} Bg^i, \rangle}{\langle d^{i-1}, Bd^{i-1} \rangle} = \frac{\|g^i\|_2^2}{\alpha^{i-1} \langle d^{i-1}, Bd^{i-1} \rangle} = \frac{\|g^i\|_2^2}{\|g^{i-1}\|_2^2}.$$

Lemma C.2.10. Under the assumptions of Lemma C.2.4,

$$\langle p^i, g\rangle \leq \langle p^{i-1}, g\rangle < 0 \text{ for all } i > 0.$$

Proof. It follows immediately from (C.21), (C.22) and (C.25) that

$$p^i = p^{i-1} - \frac{\langle g, d^{i-1}\rangle}{\langle d^{i-1}, Bd^{i-1}\rangle} d^{i-1},$$

and thus

$$\langle g, p^i\rangle = \langle g, p^{i-1}\rangle - \frac{\langle g, d^{i-1}\rangle^2}{\langle d^{i-1}, Bd^{i-1}\rangle},$$

from which it follows that $\langle g, p^i\rangle < \langle g, p^{i-1}\rangle$. The result then follows by induction, since

$$\langle g, p^1\rangle = -\frac{\|g\|_2^4}{\langle g, Bg\rangle} < 0.$$

Proof of Theorem 3.1

Firstly note that, for all $\alpha \geq 0$,

$$m_k(-\alpha g_k) = f_k - \alpha\|g_k\|_2^2 + \tfrac{1}{2}\alpha^2\langle g_k, B_k g_k\rangle. \tag{C.29}$$

If g_k is zero, the result is immediate. So suppose otherwise. In this case, there are three possibilities:

(i) the curvature $\langle g_k, B_k g_k\rangle$ is not strictly positive; in this case $m_k(-\alpha g_k)$ is unbounded from below as α increases, and hence the Cauchy point occurs on the trust-region boundary.

(ii) the curvature $\langle g_k, B_k g_k\rangle > 0$ and the minimizer of $m_k(-\alpha g_k)$ occurs at or beyond the trust-region boundary; once again, the the Cauchy point occurs on the trust-region boundary.

(iii) the curvature $\langle g_k, B_k g_k\rangle > 0$ and the minimizer of $m_k(-\alpha g_k)$, and hence the Cauchy point, occurs before the trust-region is reached.

We consider each case in turn;
Case (i). In this case, since $\langle g_k, B_k g_k\rangle \leq 0$, (C.29) gives

$$m_k(-\alpha g_k) = f_k - \alpha\|g_k\|_2^2 + \tfrac{1}{2}\alpha^2\langle g_k, B_k g_k\rangle \leq f_k - \alpha\|g_k\|_2^2 \tag{C.30}$$

for all $\alpha \geq 0$. Since the Cauchy point lies on the boundary of the trust region

$$\alpha_k^{\mathrm{C}} = \frac{\Delta_k}{\|g_k\|}. \tag{C.31}$$

Substituting this value into (C.30) gives

$$f_k - m_k(s_k^{\mathrm{C}}) \geq \|g_k\|_2^2 \frac{\Delta_k}{\|g_k\|} \geq \kappa_s\|g_k\|_2\Delta_k \geq \tfrac{1}{2}\kappa_s\|g_k\|_2\Delta_k \tag{C.32}$$

since $\|g_k\|_2 \geq \kappa_s\|g_k\|$.

Case (ii). In this case, let α_k^* be the unique minimizer of (C.29); elementary calculus reveals that

$$\alpha_k^* = \frac{\|g_k\|_2^2}{\langle g_k, B_k g_k\rangle}. \tag{C.33}$$

Since this minimizer lies on or beyond the trust-region boundary (C.31) and (C.33) together imply that

$$\alpha_k^{\mathrm{C}} \langle g_k, B_k g_k\rangle \leq \|g_k\|_2^2.$$

Substituting this last inequality in (C.29), and using (C.31) and $\|g_k\|_2 \geq \kappa_s \|g_k\|$, it follows that

$$f_k - m_k(s_k^{\mathrm{C}}) = \alpha_k^{\mathrm{C}} \|g_k\|_2^2 - \tfrac{1}{2}[\alpha_k^{\mathrm{C}}]^2 \langle g_k, B_k g_k\rangle \geq \tfrac{1}{2}\alpha_k^{\mathrm{C}}\|g_k\|_2^2 = \tfrac{1}{2}\|g_k\|_2^2 \frac{\Delta_k}{\|g_k\|} \geq \tfrac{1}{2}\kappa_s \|g_k\|_2 \Delta_k.$$

Case (iii). In this case, $\alpha_k^{\mathrm{C}} = \alpha_k^*$, and (C.29) becomes

$$f_k - m_k(s_k^{\mathrm{C}}) = \frac{\|g_k\|_2^4}{\langle g_k, B_k g_k\rangle} - \tfrac{1}{2}\frac{\|g_k\|_2^4}{\langle g_k, B_k g_k\rangle} = \tfrac{1}{2}\frac{\|g_k\|_2^4}{\langle g_k, B_k g_k\rangle} \geq \tfrac{1}{2}\frac{\|g_k\|_2^2}{1 + \|B_k\|_2},$$

where

$$|\langle g_k, B_k g_k\rangle| \leq \|g_k\|_2^2 \|B_k\|_2 \leq \|g_k\|_2^2 (1 + \|B_k\|_2)$$

because of the Cauchy-Schwarz inequality.

The result follows since it is true in each of the above three possible cases. Note that the "1+" is only needed to cover case where $B_k = 0$, and that in this case, the "min" in the theorem might actually be replaced by $\kappa_s \Delta_k$.

Proof of Corollary 3.2

Immediate from Theorem 3.1 and the requirement that $m_k(s_k) \leq m_k(s_k^{\mathrm{C}})$.

Proof of Lemma 3.3

The generalized mean-value theorem gives that

$$f(x_k + s_k) = f(x_k) + \langle s_k, \nabla_x f(x_k)\rangle + \tfrac{1}{2}\langle s_k, \nabla_{xx} f(\xi_k) s_k\rangle$$

for some ξ_k in the segment $[x_k, x_k + s_k]$. Thus

$$\begin{aligned} |f(x_k + s_k) - m_k(s_k)| &= \tfrac{1}{2}|\langle s_k, H(\xi_k)s_k\rangle - \langle s_k, B_k s_k\rangle| \leq \tfrac{1}{2}|\langle s_k, H(\xi_k)s_k\rangle| + \tfrac{1}{2}|\langle s_k, B_k s_k\rangle| \\ &\leq \tfrac{1}{2}(\kappa_h + \kappa_b)\|s_k\|_2^2 \leq \tfrac{1}{2}\kappa_l^2(\kappa_h + \kappa_b)\|s_k\|^2 \leq \kappa_d \Delta_k^2 \end{aligned}$$

using the triangle and Cauchy-Schwarz inequalities.

Proof of Lemma 3.4

By definition,

$$1 + \|B_k\|_2 \leq \kappa_h + \kappa_b,$$

and hence for any radius satisfying the given (first) bound,

$$\kappa_s \Delta_k \leq \frac{\|g_k\|_2}{\kappa_h + \kappa_b} \leq \frac{\|g_k\|_2}{1 + \|B_k\|_2}.$$

As a consequence, Corollary 3.2 gives that

$$f_k - m_k(s_k) \geq \tfrac{1}{2}\|g_k\|_2 \min\left[\frac{\|g_k\|_2}{1+\|B_k\|_2}, \kappa_s \Delta_k\right] = \tfrac{1}{2}\kappa_s \|g_k\|_2 \Delta_k. \tag{C.34}$$

But then Lemma 3.3 and the assumed (second) bound on the radius gives that

$$|\rho_k - 1| = \left| \frac{f(x_k + s_k) - m_k(s_k)}{f_k - m_k(s_k)} \right| \leq 2 \frac{\kappa_d \Delta_k^2}{\kappa_s \|g_k\|_2 \Delta_k} = \frac{2\kappa_d}{\kappa_s} \frac{\Delta_k}{\|g_k\|_2} \leq 1 - \eta_v. \tag{C.35}$$

Therefore, $\rho_k \geq \eta_v$ and the iteration is very successful.

Proof of Lemma 3.5

Suppose otherwise that Δ_k can become arbitrarily small. In particular, assume that iteration k is the first such that

$$\Delta_{k+1} \leq \kappa_\epsilon. \tag{C.36}$$

Then since the radius for the previous iteration must have been larger, the iteration was unsuccessful, and thus $\gamma_d \Delta_k \leq \Delta_{k+1}$. Hence

$$\Delta_k \leq \epsilon \min\left(\frac{1}{\kappa_s(\kappa_h + \kappa_b)}, \frac{\kappa_s(1-\eta_v)}{2\kappa_d}\right) \leq \|g_k\| \min\left(\frac{1}{\kappa_s(\kappa_h + \kappa_b)}, \frac{\kappa_s(1-\eta_v)}{2\kappa_d}\right).$$

But this contradicts the assertion of Lemma 3.4 that the k-th iteration must be very successful.

Proof of Lemma 3.6

The mechanism of the algorithm ensures that $x_* = x_{k_0+1} = x_{k_0+j}$ for all $j > 0$, where k_0 is the index of the last successful iterate. Moreover, since all iterations are unsuccessful for sufficiently large k, the sequence $\{\Delta_k\}$ converges to zero. If $\|g_{k_0+1}\| > 0$, Lemma 3.4 then implies that there must be a successful iteration of index larger than k_0, which is impossible. Hence $\|g_{k_0+1}\| = 0$.

Proof of Theorem 3.7

Lemma 3.6 shows that the result is true when there are only a finite number of successful iterations. So it remains to consider the case where there are an infinite number of successful iterations. Let $\mathcal{S}$ be the index set of successful iterations. Now suppose that

$$\|g_k\| \geq \epsilon \tag{C.37}$$

for some $\epsilon > 0$ and all k, and consider a successful iteration of index k. The fact that k is successful, Corollary 3.2, Lemma 3.5, and the assumption (C.37) give that

$$f_k - f_{k+1} \geq \eta_s [f_k - m_k(s_k)] \geq \delta_\epsilon \stackrel{\text{def}}{=} \tfrac{1}{2}\eta_s \epsilon \min\left[\frac{\epsilon}{1+\kappa_b}, \kappa_s \kappa_\epsilon\right]. \tag{C.38}$$

Summing now over all successful iterations from 0 to k, it follows that

$$f_0 - f_{k+1} = \sum_{\substack{j=0 \\ j \in \mathcal{S}}}^{k} [f_j - f_{j+1}] \geq \sigma_k \delta_\epsilon,$$

where σ_k is the number of successful iterations up to iteration k. But since there are infinitely many such iterations, it must be that

$$\lim_{k \to \infty} \sigma_k = +\infty.$$

Thus (C.37) can only be true if f_{k+1} is unbounded from below, and conversely, if f_{k+1} is bounded from below, (C.37) must be false, and there is a subsequence of the $\|g_k\|$ converging to zero.

Proof of Corollary 3.8

Suppose otherwise that f_k is bounded from below, and that there is a subsequence of successful iterates, indexed by $\{t_i\} \subseteq \mathcal{S}$, such that

$$\|g_{t_i}\| \geq 2\epsilon > 0 \tag{C.39}$$

for some $\epsilon > 0$ and for all i. Theorem 3.7 ensures the existence, for each t_i, of a first successful iteration $\ell_i > t_i$ such that $\|g_{\ell_i}\| < \epsilon$. That is to say that there is another subsequence of $\mathcal{S}$ indexed by $\{\ell_i\}$ such that

$$\|g_k\| \geq \epsilon \;\text{ for }\; t_i \leq k < \ell_i \;\text{ and }\; \|g_{\ell_i}\| < \epsilon. \tag{C.40}$$

We now restrict our attention to the subsequence of successful iterations whose indices are in the set

$$\mathcal{K} \stackrel{\text{def}}{=} \{k \in \mathcal{S} \mid t_i \leq k < \ell_i\},$$

where t_i and ℓ_i belong to the two subsequences defined above.

The subsequences $\{t_i\}$, $\{\ell_i\}$ and $\mathcal{K}$ are all illustrated in Figure C.1, where, for simplicity, it is assumed that all iterations are successful. In this figure, we have marked position j in each of the subsequences represented in abscissa when j belongs to that subsequence. Note in this example that $\ell_0 = \ell_1 = \ell_2 = \ell_3 = \ell_4 = \ell_5 = 8$, which we indicated by arrows from $t_0 = 0$, $t_1 = 1$, $t_2 = 2$, $t_3 = 3$, $t_4 = 4$ and $t_5 = 7$ to $k = 9$, and so on.

As in the previous proof, it immediately follows that

$$f_k - f_{k+1} \geq \eta_s[f_k - m_k(s_k)] \geq \tfrac{1}{2}\eta_s \epsilon \min\left[\frac{\epsilon}{1+\kappa_b}, \kappa_s \Delta_k\right] \tag{C.41}$$

holds for all $k \in \mathcal{K}$ because of (C.40). Hence, since $\{f_k\}$ is, by assumption, bounded from below, the left-hand side of (C.41) must tend to zero when k tends to infinity, and thus that

$$\lim_{\substack{k \to \infty \\ k \in \mathcal{K}}} \Delta_k = 0.$$

As a consequence, the second term dominates in the minimum of (C.41) and it follows that, for $k \in \mathcal{K}$ sufficiently large,

$$\Delta_k \leq \frac{2}{\epsilon \eta_s \kappa_s}[f_k - f_{k+1}].$$

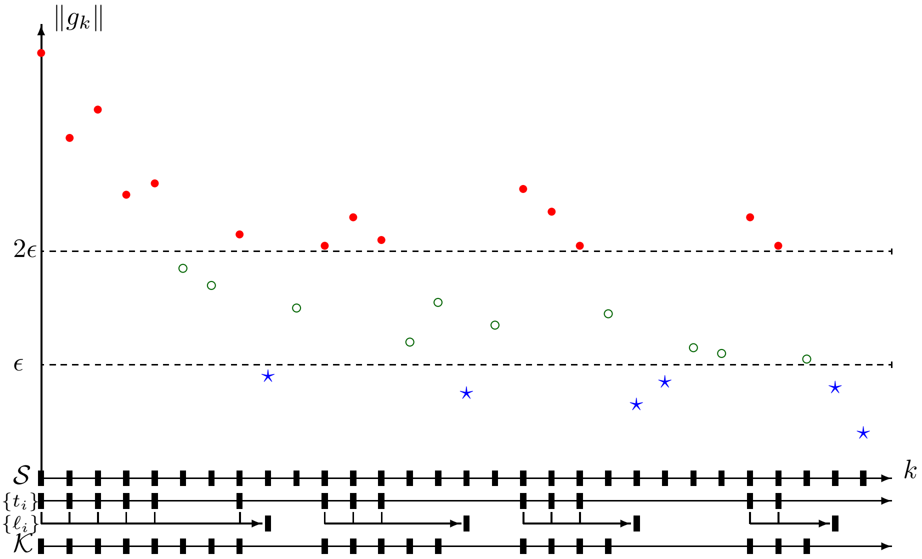

Figure C.1: The subsequences of the proof of Corollary 3.8

We then deduce from this bound that, for i sufficiently large,

$$\|x_{t_i} - x_{\ell_i}\| \le \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{\ell_i-1} \|x_j - x_{j+1}\| \le \sum_{\substack{j=t_i \\ j\in\mathcal{K}}}^{\ell_i-1} \Delta_j \le \frac{2}{\epsilon\eta_s\kappa_s}[f_{t_i} - f_{\ell_i}]. \tag{C.42}$$

But, because $\{f_k\}$ is monotonic and, by assumption, bounded from below, the right-hand side of (C.42) must converge to zero. Thus $\|x_{t_i} - x_{\ell_i}\|$ tends to zero as i tends to infinity, and hence, by continuity, $\|g_{t_i} - g_{\ell_i}\|$ also tend to zero. However this is impossible because of the definitions of $\{t_i\}$ and $\{\ell_i\}$, which imply that $\|g_{t_i} - g_{\ell_i}\| \ge \epsilon$. Hence, no subsequence satisfying (C.39) can exist.

Proof of Theorem 3.9

The constraint $\|s\|_2 \le \Delta$ is equivalent to

$$\tfrac{1}{2}\Delta^2 - \tfrac{1}{2}\langle s, s\rangle \ge 0. \tag{C.43}$$

Applying Theorem 1.9 to the problem of minimizing $q(s)$ subject to (C.43) gives

$$g + Bs_* = -\lambda_* s_* \tag{C.44}$$

for some Lagrange multiplier $\lambda_* \ge 0$ for which either $\lambda_* = 0$ or $\|s_*\|_2 = \Delta$ (or both). It remains to show that $B + \lambda_* I$ is positive semi-definite.

If s_* lies in the interior of the trust-region, necessarily $\lambda_* = 0$, and Theorem 1.10 implies that $B + \lambda_* I = B$ must be positive semi-definite. Likewise if $\|s_*\|_2 = \Delta$ and $\lambda_* = 0$, it follows from Theorem 1.10 that necessarily $\langle v, Bv\rangle \ge 0$ for all $v \in \mathcal{N}_+ = \{v | \langle s_*, v\rangle \ge 0\}$. If $v \notin \mathcal{N}_+$, then $-v \in \mathcal{N}_+$, and thus $\langle v, Bv\rangle \ge 0$ for all v. Thus the only outstanding case is where $\|s_*\|_2 = \Delta$ and

$\lambda_* > 0$. In this case, Theorem 1.10 shows that $\langle v, (B + \lambda_* I)v\rangle \geq 0$ for all $v \in \mathcal{N}_+ = \{v | \langle s_*, v\rangle = 0\}$, so it remains to consider $\langle v, Bv\rangle$ when $\langle s_*, v\rangle \neq 0$.

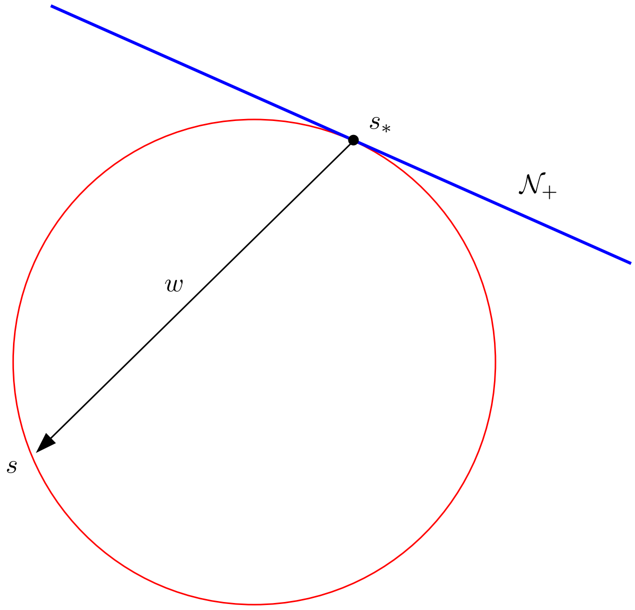

Figure C.2: Construction of "missing" directions of positive curvature.

Let s be any point on the boundary of the trust-region, and let $w = s - s_*$, as in Figure C.2. Then

$$-\langle w, s_*\rangle = \langle s_* - s, s_*\rangle = \tfrac{1}{2}\langle s_* - s, s_* - s\rangle = \tfrac{1}{2}\langle w, w\rangle \tag{C.45}$$

since $\|s\|_2 = \Delta = \|s_*\|_2$. Combining this with (C.44) gives

$$q(s) - q(s_*) = \langle w, g + Bs_*\rangle + \tfrac{1}{2}\langle w, Bw\rangle = -\lambda_*\langle w, s_*\rangle + \tfrac{1}{2}\langle w, Bw\rangle = \tfrac{1}{2}\langle w, (B + \lambda_* I)w\rangle, \tag{C.46}$$

and thus necessarily $\langle w, (B + \lambda_* I)w\rangle \geq 0$ since s_* is a global minimizer. It is easy to show that

$$s = s_* - 2\frac{\langle s_*, v\rangle}{\langle v, v\rangle}v$$

lies on the trust-region boundary, and thus for this s, w is parallel to v from which it follows that $\langle v, (B + \lambda_* I)v\rangle \geq 0$.

When $B + \lambda_* I$ is positive definite, $s_* = -(B + \lambda_* I)^{-1}g$. If this point is on the trust-region boundary, while s is any value in the trust-region, (C.45) and (C.46) become $-\langle w, s_*\rangle \geq \frac{1}{2}\langle w, w\rangle$ and $q(s) \geq q(s_*) + \frac{1}{2}\langle w, (B + \lambda_* I)w\rangle$ respectively. Hence, $q(s) > q(s_*)$ for any $s \neq s_*$. If s_* is interior, $\lambda_* = 0$, B is positive definite, and thus s_* is the unique unconstrained minimizer of $q(s)$.

Newton's method for the secular equation (Part 3)

Recall that the Newton correction at λ is $-\phi(\lambda)/\phi'(\lambda)$. Since

$$\phi(\lambda) = \frac{1}{\|s(\lambda)\|_2} - \frac{1}{\Delta} = \frac{1}{(\langle s(\lambda), s(\lambda)\rangle)^{\frac{1}{2}}} - \frac{1}{\Delta},$$

it follows, on differentiating, that

$$\phi'(\lambda) = -\frac{\langle s(\lambda), \nabla_\lambda s(\lambda)\rangle}{(\langle s(\lambda), s(\lambda)\rangle)^{\frac{3}{2}}} = -\frac{\langle s(\lambda), \nabla_\lambda s(\lambda)\rangle}{\|s(\lambda)\|_2^3}.$$

In addition, on differentiating the defining equation

$$(B + \lambda I)s(\lambda) = -g,$$

it must be that

$$(B + \lambda I)\nabla_\lambda s(\lambda) + s(\lambda) = 0.$$

Notice that, rather than the value of $\nabla_\lambda s(\lambda)$, merely the numerator

$$\langle s(\lambda), \nabla_\lambda s(\lambda)\rangle = -\langle s(\lambda), (B + \lambda I)(\lambda)^{-1} s(\lambda)\rangle$$

is required in the expression for $\phi'(\lambda)$. Given the factorization $B + \lambda I = L(\lambda)L^T(\lambda)$, the simple relationship

$$\langle s(\lambda), (B + \lambda I)^{-1} s(\lambda)\rangle = \langle s(\lambda), L^{-T}(\lambda)L^{-1}(\lambda)s(\lambda)\rangle = \langle L^{-1}(\lambda)s(\lambda), L^{-1}(\lambda)s(\lambda)\rangle = \|w(\lambda)\|_2^2$$

where $L(\lambda)w(\lambda) = s(\lambda)$ then justifies the Newton step.

Proof of Theorem 3.10

We first show that

$$\langle d^i, d^j\rangle = \frac{\|g^i\|_2^2}{\|g^j\|_2^2}\|d^j\|_2^2 > 0 \tag{C.47}$$

for all $0 \le j \le i \le k$. For any i, (C.47) is trivially true for $j = i$. Suppose it is also true for all $i \le l$. Then, the update for d^{l+1} gives

$$d^{l+1} = -g^{l+1} + \frac{\|g^{l+1}\|_2^2}{\|g^l\|_2^2} d^l.$$

Forming the inner product with d^j, and using the fact that $\langle d^j, g^{l+1}\rangle = 0$ for all $j = 0, \ldots, l$, and (C.47) when $j = l$, reveals

$$\langle d^{l+1}, d^j\rangle = -\langle g^{l+1}, d^j\rangle + \frac{\|g^{l+1}\|_2^2}{\|g^l\|_2^2}\langle d^l, d^j\rangle = \frac{\|g^{l+1}\|_2^2}{\|g^l\|_2^2}\frac{\|g^l\|_2^2}{\|g^j\|_2^2}\|d^j\|_2^2 = \frac{\|g^{l+1}\|_2^2}{\|g^j\|_2^2}\|d^j\|_2^2 > 0.$$

Thus (C.47) is true for $i \le l + 1$, and hence for all $0 \le j \le i \le k$.

We now have from the algorithm that

$$s^i = s^0 + \sum_{j=0}^{i-1} \alpha^j d^j = \sum_{j=0}^{i-1} \alpha^j d^j$$

as, by assumption, $s^0 = 0$. Hence

$$\langle s^i, d^i \rangle = \left\langle \sum_{j=0}^{i-1} \alpha^j d^j, d^i \right\rangle = \sum_{j=0}^{i-1} \alpha^j \langle d^j, d^i \rangle > 0 \tag{C.48}$$

as each $\alpha^j > 0$, which follows from the definition of α^j, since $\langle d^j, Hd^j \rangle > 0$, and from relationship (C.47). Hence

$$\begin{aligned} \|s^{i+1}\|_2^2 &= \langle s^{i+1}, s^{i+1} \rangle = \langle s^i + \alpha^i d^i, s^i + \alpha^i d^i \rangle \\ &= \langle s^i, s^i \rangle + 2\alpha^i \langle s^i, d^i \rangle + \alpha^{i\,2} \langle d^i, d^i \rangle > \langle s^i, s^i \rangle = \|s^i\|_2^2 \end{aligned}$$

follows directly from (C.48) and $\alpha^i > 0$ which is the required result.

Proof of Theorem 3.11

The proof is elementary but rather complicated. See

Y. Yuan, "On the truncated conjugate-gradient method", *Mathematical Programming*, **87** (2000) 561:573

for full details.

Proof of Theorem 4.1

The proof of this is sufficiently non trivial that early proofs were incomplete. For a correct version, see

J. M. Borwein, "Necessary and sufficient conditions for quadratic minimality", *Numerical Functional Analysis and Optimization*, **5** (1982) 127:140.

Proof of Theorem 4.4

We prove this first, and then deduce Theorems 4.2 and 4.3—we suppose throughout that A is of full rank. The result is actually a trivial consequence of the definition of second-order sufficiency. For if S denotes a basis for the null-space of A, any vector lying in this null space may be expressed as $s = Ss_N$. Since $\langle s, Hs \rangle > 0$ for all such vectors, $\langle s_N, S^T HSs_N \rangle > 0$ for all nonzero s_N, which is true if and only if $S^T HS$ is positive definite.

Proof of Theorem 4.2

The proof is straightforward, but depends heavily on three consequences of a classical result, namely Sylvester (1814–1897)'s law of inertia. The inertia of a symmetric matrix K is the triplet

$$\mathrm{In}(K) \stackrel{\text{def}}{=} (k_+, k_-, k_0),$$

where k_+, k_- and k_0 denote respectively the numbers of (strictly) positive, negative and zero eigenvalues of K.

Theorem C.4.1. (Sylvester's law of inertia). Let K be a given symmetric matrix, and S any non-singular matrix. Then

$$\mathrm{In}(K) = \mathrm{In}(S^T K S).$$

This has the immediate consequences

Lemma C.4.2. Let K be a given symmetric matrix. Then symmetric (row-and-column) permutations of K do not change its inertia.

Proof: This follows immediately from Theorem C.4.1 by picking S as the required permutation matrix. □

Lemma C.4.3. Suppose that C and F are symmetric, and additionally C is invertible. Then

$$\mathrm{In}\begin{pmatrix} C & E^T \\ E & F \end{pmatrix} = \mathrm{In}(C) + \mathrm{In}(F - EC^{-1}E^T). \tag{C.49}$$

Proof: Since

$$M \stackrel{\text{def}}{=} \begin{pmatrix} C & E^T \\ E & F \end{pmatrix} = \begin{pmatrix} I & 0 \\ EC^{-1} & I \end{pmatrix} \begin{pmatrix} C & 0 \\ 0 & F - EC^{-1}E^T \end{pmatrix} \begin{pmatrix} I & C^{-1}E^T \\ 0 & I \end{pmatrix},$$

the required result (C.49) follows from Theorem C.4.1, as the middle matrix in the above decomposition of M is block diagonal and the outer pair of matrices are non-singular transposes of each other. □

Lemma C.4.4. Suppose that E is p by p and symmetric. Then

$$\mathrm{In}\begin{pmatrix} E & I \\ I & 0 \end{pmatrix} = (p, p, 0). \tag{C.50}$$

Proof: To see this, let E have spectral decomposition QDQ^T, where Q is orthonormal and D is diagonal with diagonal entries d_i, $i = 1, \ldots, p$. Then

$$\begin{pmatrix} Q & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} D & I \\ I & 0 \end{pmatrix} \begin{pmatrix} Q^T & 0 \\ 0 & Q^T \end{pmatrix} = \begin{pmatrix} QDQ^T & QQ^T \\ QQ^T & 0 \end{pmatrix} = \begin{pmatrix} E & I \\ I & 0 \end{pmatrix},$$

so

$$\mathrm{In}\begin{pmatrix} E & I \\ I & 0 \end{pmatrix} = \mathrm{In}\begin{pmatrix} D & I \\ I & 0 \end{pmatrix}$$

using Theorem C.4.1. But the eigenvalues λ of the latter satisfy

$$\begin{pmatrix} D & I \\ I & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}$$

from which it follows trivially that $\lambda = -\frac{1}{2}d_i \pm \frac{1}{2}\sqrt{d_i^2 + 4}$. Since one of these is strictly positive, and the other is strictly negative for each $i = 1, \ldots p$, (C.50) is true. □

Armed with these lemmata, to prove the theorem let the m by n matrix A have an LQ factors

$$AQ \equiv A(Y \;\; S) = (L \;\; 0),$$

where L is (lower triangular and) invertible, Q is orthonormal and Y and S are columns of Q. Notice that S gives a basis for the null-space of A since $AS = 0$. Then

$$\begin{pmatrix} Q^T & 0 \\ 0 & L^{-1} \end{pmatrix} \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} Q & 0 \\ 0 & L^{-T} \end{pmatrix} = \begin{pmatrix} Q^T H Q & Q^T A^T L^{-T} \\ L^{-1} A Q & 0 \end{pmatrix} = \begin{pmatrix} Y^T H Y & Y^T H S & I \\ S^T H Y & S^T H S & 0 \\ I & 0 & 0 \end{pmatrix}.$$

But Theorem 4.4 shows that $\mathrm{In}(S^T H S) = (n - m, 0, 0)$. Hence Theorem C.4.1 and Lemmas C.4.2–C.4.4 give

$$\begin{aligned}
\mathrm{In}(K) & \stackrel{\text{Thm C.4.1}}{=} \mathrm{In} \begin{pmatrix} Y^T H Y & Y^T H S & I \\ S^T H Y & S^T H S & 0 \\ I & 0 & 0 \end{pmatrix} \\
& \stackrel{\text{Lem C.4.2}}{=} \mathrm{In} \begin{pmatrix} Y^T H Y & I & Y^T H S \\ I & 0 & 0 \\ S^T H Y & 0 & S^T H S \end{pmatrix} \\
& \stackrel{\text{Lem C.4.4}}{=} \mathrm{In} \begin{pmatrix} Y^T H Y & I \\ I & 0 \end{pmatrix} + \mathrm{In} \left(S^T H S - (S^T H Y \; 0) \begin{pmatrix} Y^T H Y & I \\ I & 0 \end{pmatrix}^{-1} \begin{pmatrix} Y^T H S \\ 0 \end{pmatrix} \right) \\
& \stackrel{\text{Lem C.4.3}}{=} (m, m, 0) + \mathrm{In} \left(S^T H S - (S^T H Y \; 0) \begin{pmatrix} 0 & I \\ I & -Y^T H Y \end{pmatrix} \begin{pmatrix} Y^T H S \\ 0 \end{pmatrix} \right) \\
& = (m, m, 0) + \mathrm{In}(S^T H S) \\
& \stackrel{\text{Thm 4.4}}{=} (m, m, 0) + (n - m, 0, 0) = (n, m, 0)
\end{aligned}$$

as required.

Proof of Theorem 4.3

Since by assumption H, and *a fortiori*, $AH^{-1}A^T$ are non-singular, let $\mathrm{In}(H) = (n - h_-, h_-, 0)$ and $\mathrm{In}(AH^{-1}A^T) = (m - s_-, s_-, 0)$. The result then follows immediately by applying Theorem 4.2 and Lemma C.4.3 to K to find

$$(n, m, 0) = \mathrm{In}(K) = \mathrm{In}(H) + \mathrm{In}(-AH^{-1}A^T) = (n - h_-, h_-, 0) + (s_-, m - s_-, 0),$$

and hence that $h_- = s_-$.

Proof of Theorem 5.1

Denote the left generalized inverse of $A^T(x)$ by

$$A^+(x) = \left(A(x) A^T(x)\right)^{-1} A(x)$$

at any point for which $A(x)$ is full rank. Since, by assumption, $A(x_*)$ is full rank, these generalized inverses exists, and are bounded and continuous in some open neighbourhood of x_*.

Now let

$$y_k = -\frac{c(x_k)}{\mu_k}$$

as well as

$$y_* = A^+(x_*)g(x_*).$$

It then follows from the inner-iteration termination test

$$\|g(x_k) - A^T(x_k)y_k\| \le \epsilon_k \tag{C.51}$$

and the continuity of $A^+(x_k)$ that

$$\|A^+(x_k)g(x_k) - y_k\|_2 = \left\|A^+(x_k)\left(g(x_k) - A^T(x_k)y_k\right)\right\|_2 \le 2\|A^+(x_*)\|_2\epsilon_k.$$

Then

$$\|y_k - y_*\|_2 \le \|A^+(x_*)g(x_*) - A^+(x_k)g(x_k)\|_2 + \|A^+(x_k)g(x_k) - y_k\|_2$$

which implies that $\{y_k\}$ converges to y_*. In addition, continuity of the gradients and (C.51) implies that

$$g(x_*) - A^T(x_*)y_* = 0,$$

while the fact that $c(x_k) = -\mu_k y_k$ with bounded y_k implies that

$$c(x_*) = 0.$$

Hence (x_*, y_*) satisfies the first-order optimality conditions.

Proof of Theorem 5.2

A formal proof is given by

W. Murray, "Analytical expressions for eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions", *J. Optimization Theory and Applics*, **7** (1971) 189:196.

By way of a sketch, let $Q(x)$ and $S(x)$ be orthonormal bases for the range- and null-spaces of $A(x)$. As we have shown, the required Hessian may be expressed (in decreasing terms of asymptotic dominance) as

$$\nabla_{xx}\Phi(x,\mu) = A_{\mathcal{A}}^T(x)A_{\mathcal{A}}(x)/\mu + H(x, y(x,\mu)).$$

Since the eigenvalues of $\nabla_{xx}\Phi(x,\mu)$ are not affected by orthonormal transformations, on pre- and post-multiplying $\nabla_{xx}\Phi(x,\mu)$ by $(Q(x) \;\; S(x))$ and its transpose, we see that the required eigenvalues are those of

$$\begin{pmatrix} Q(x)^T A^T(x)A(x)Q(x)/\mu + Q(x)^T H(x,y(x,\mu))Q(x) & Q(x)^T H(x,y(x,\mu))S(x) \\ S(x)^T H(x,y(x,\mu))Q(x) & S(x)^T H(x,y(x,\mu))S(x) \end{pmatrix} + O(\mu), \tag{C.52}$$

where we have use the relationship $A(x)S(x) = 0$. The dominant eigenvalues are those arising from the 1,1 block of (C.52), and these are those of $Q(x)^T A^T(x)A(x)Q(x)/\mu$ with an $O(1)$ error—these are the same as those of

$$A(x)Q(x)Q(x)^T A^T(x)\mu = A(x)A^T(x)/\mu$$

as $Q(x)Q^T(x) = I$ and because the non-zero eigenvalues of B^TB and BB^T agree for any (rectangular or otherwise) matrix B. Since the remaining eigenvalues must occur for eigenvectors orthogonal to those giving the 1,1 block, they will asymptotically be those of the 2,2 block, and thus those of $S(x)^T H(x,y(x,\mu))S(x)$ with an $O(\mu)$ term.

Proof of Theorem 5.3

The proof of convergence of y_k to $y_* \stackrel{\text{def}}{=} A^+(x_*)g(x_*)$ for which $g(x_*) = A^T(x_*)y_*$ is exactly as for Theorem 5.1. For the second part of the theorem, the definition of y_k and the triangle inequality gives

$$\|c(x_k)\| = \mu_k\|u_k - y_k\| \leq \mu_k\|y_k - y_*\| + \mu_k\|u_k - y_*\|.$$

the first term on the right-hand side converges to zero as y_k approaches y_* with bounded μ_k, while the second term has the same limit because of the assumptions made. Hence $c(x_*) = 0$, and (x_*, y_*) satisfies the first-order optimality conditions.

Proof of Theorem 6.1

Let $\mathcal{A} = \mathcal{A}(x_*)$, and $\mathcal{I} = \{1, \ldots, m\} \setminus \mathcal{A}$ be the indices of constraints that are active and inactive at x_*. Furthermore let subscripts $_\mathcal{A}$ and $_\mathcal{I}$ denote the rows of matrices/vectors whose indices are indexed by these sets. Denote the left generalized inverse of $A^T_\mathcal{A}(x)$ by

$$A^+_\mathcal{A}(x) = \left(A_\mathcal{A}(x)A^T_\mathcal{A}(x)\right)^{-1} A_\mathcal{A}(x)$$

at any point for which $A_\mathcal{A}(x)$ is full rank. Since, by assumption, $A_\mathcal{A}(x_*)$ is full rank, these generalized inverses exists, and are bounded and continuous in some open neighbourhood of x_*.

Now let

$$(y_k)_i = \frac{\mu_k}{c_i(x_k)}$$

for $i = 1, \ldots, m$, as well as

$$(y_*)_\mathcal{A} = A^+_\mathcal{A}(x_*)g(x_*)$$

and $(y_*)_\mathcal{I} = 0$. If $\mathcal{I} \neq \emptyset$, then

$$\|(y_k)_\mathcal{I}\|_2 \leq 2\mu_k\sqrt{|\mathcal{I}|}/\min_{i\in\mathcal{I}}|c_i(x_*)| \tag{C.53}$$

for all sufficiently large k. It then follows from the inner-iteration termination test that

$$\begin{aligned} \|g(x_k) - A^T_\mathcal{A}(x_k)(y_k)_\mathcal{A}\|_2 \leq & \quad \|g(x_k) - A^T(x_k)y_k\|_2 + \|A^T_\mathcal{I}(x_k)(y_k)_\mathcal{I}\|_2 \\ \leq & \quad \bar{\epsilon}_k \stackrel{\text{def}}{=} \epsilon_k + \mu_k\frac{2\sqrt{|\mathcal{I}|}\|A_\mathcal{I}\|_2}{\min_{i\in\mathcal{I}}|c_i(x_*)|}. \end{aligned} \tag{C.54}$$

Hence

$$\|A^+_\mathcal{A}(x_k)g(x_k) - (y_k)_\mathcal{A}\|_2 = \|A^+_\mathcal{A}(x_k)(g(x_k) - A^T_\mathcal{A}(x_k)(y_k)_\mathcal{A})\|_2 \leq 2\|A^+_\mathcal{A}(x_*)\|_2\bar{\epsilon}_k.$$

Then

$$\|(y_k)_\mathcal{A} - (y_*)_\mathcal{A}\|_2 \leq \|A^+_\mathcal{A}(x_*)g(x_*) - A^+_\mathcal{A}(x_k)g(x_k)\|_2 + \|A^+_\mathcal{A}(x_k)g(x_k) - (y_k)_\mathcal{A}\|_2$$

which, in combination with (C.53) and convergence of x_k, implies that $\{y_k\}$ converges to y_*. In addition, continuity of the gradients and (C.54) implies that

$$g(x_*) - A^T(x_*)y_* = 0$$

while the fact that $c(x_k) > 0$ for all k, the definition of y_k and y_* (and the implication that $c_i(x_k)(y_k)_i = \mu_k$) shows that $c(x_*) \geq 0$, $y_* \geq 0$ and $c_i(x_*)(y_*)_i = 0$. Hence (x_*, y_*) satisfies the first-order optimality conditions.

Proof of Theorem 6.2

This is essentially the same as that for Theorem 5.2, as before given formally by

W. Murray, "Analytical expressions for eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions", *J. Optimization Theory and Applics*, **7** (1971) 189:196.

By way of a sketch, let $Q(x)$ and $S(x)$ be orthonormal bases for the range- and null-spaces of $A_{\mathcal{A}(x_*)}(x)$, and let $A_{\mathcal{I}}(x)$ be the matrix whose rows are $\{a_i^T(x)\}_{i \notin \mathcal{A}(x_*)}$. As we have shown, the required Hessian may be expressed (in decreasing terms of asymptotic dominance) as

$$\nabla_{xx}\Phi(x,\mu) = A_{\mathcal{A}}^T(x)Y_{\mathcal{A}}^2(x,\mu)A_{\mathcal{A}}(x)/\mu + H(x,y(x,\mu)) + \mu A_{\mathcal{I}}^T(x)C_{\mathcal{I}}^{-2}(x)A_{\mathcal{I}}(x).$$

Since the eigenvalues of $\nabla_{xx}\Phi(x,\mu)$ are not affected by orthonormal transformations, on pre- and post-multiplying $\nabla_{xx}\Phi(x,\mu)$ by $(Q(x) \;\; S(x))$ and its transpose, we see that the required eigenvalues are those of

$$\begin{pmatrix} Q(x)^T A_{\mathcal{A}}^T(x)Y_{\mathcal{A}}^2(x,\mu)A_{\mathcal{A}}(x)Q(x)/\mu + Q(x)^T H(x,y(x,\mu))Q(x) & Q(x)^T H(x,y(x,\mu))S(x) \\ S(x)^T H(x,y(x,\mu))Q(x) & S(x)^T H(x,y(x,\mu))S(x) \end{pmatrix} + O(\mu), \tag{C.55}$$

where we have use the relationship $A(x)S(x) = 0$. The dominant eigenvalues are those arising from the 1,1 block of (C.55), and these are those of $Q(x)^T A_{\mathcal{A}}^T(x)Y_{\mathcal{A}}^2(x,\mu)A_{\mathcal{A}}(x)Q(x)/\mu$ with an $O(1)$ error—these are the same as those of

$$Y_{\mathcal{A}}(x,\mu)A_{\mathcal{A}}(x)Q(x)Q(x)^T A_{\mathcal{A}}^T(x)Y_{\mathcal{A}}(x,\mu)/\mu = Y_{\mathcal{A}}(x,\mu)A_{\mathcal{A}}(x)A_{\mathcal{A}}^T(x)Y_{\mathcal{A}}(x,\mu)/\mu$$

as $Q(x)Q^T(x) = I$ and because the non-zero eigenvalues of B^TB and BB^T agree for any (rectangular or otherwise) matrix B. Since the remaining eigenvalues must occur for eigenvectors orthogonal to those giving the 1,1 block, they will asymptotically be those of the 2,2 block, and thus those of $S(x)^T H(x,y(x,\mu))S(x)$ with an $O(\mu)$ term.

Proof of Theorem 6.3

The proof of this result is elementary, but rather long and involved. See

N. Gould, D. Orban, A. Sartenaer and Ph. L. Toint, "Superlinear convergence of primal-dual interior point algorithms for nonlinear programming", *SIAM J. Optimization*, **11**(4) (2001) 974:1002

for full details.

Proof of Theorem 7.1

The SQP search direction s_k and its associated Lagrange multiplier estimates y_{k+1} satisfy

$$B_k s_k - A_k^T y_{k+1} = -g_k \tag{C.56}$$

and

$$A_k s_k = -c_k. \tag{C.57}$$

Pre-multiplying (C.56) by s_k and using (C.57) gives that

$$\langle s_k, g_k\rangle = -\langle s_k, B_k s_k\rangle + \langle s_k, A_k^T y_{k+1}\rangle = -\langle s_k, B_k s_k\rangle - \langle c_k, y_{k+1}\rangle. \tag{C.58}$$

Likewise (C.57) gives

$$\frac{1}{\mu_k}\langle s_k, A_k^T c_k\rangle = -\frac{\|c_k\|_2^2}{\mu_k}. \tag{C.59}$$

Combining (C.58) and (C.59), and using the positive definiteness of B_k, the Cauchy-Schwarz inequality and the fact that $s_k \neq 0$ if x_k is not critical, yields

$$\begin{aligned}\langle s_k, \nabla_x \Phi(x_k)\rangle = & \left\langle s_k, g_k + \frac{1}{\mu_k} A_k^T c_k \right\rangle = -\langle s_k, B_k s_k\rangle - \langle c_k, y_{k+1}\rangle - \frac{\|c_k\|_2^2}{\mu_k} \\ < & -\|c_k\|_2 \left(\frac{\|c_k\|_2}{\mu_k} - \|y_{k+1}\|_2 \right) \leq 0\end{aligned}$$

because of the required bound on μ_k.

Proof of Theorem 7.2

The proof is slightly complicated as it uses the calculus of non-differentiable functions. See Theorem 14.3.1 in

R. Fletcher, "Practical Methods of Optimization", Wiley (1987, 2nd edition),

where the converse result, that if x_* is an isolated local minimizer of $\Phi(x, \rho)$ for which $c(x_*) = 0$ then x_* solves the given nonlinear program so long as ρ is sufficiently large, is also given. Moreover, Fletcher showns (Theorem 14.3.2) that x_* cannot be a local minimizer of $\Phi(x, \rho)$ when $\rho < \|y_*\|_D$.

Proof of Theorem 7.3

For small steps α, Taylor's theorem applied separately to f and c, along with (C.57), gives that

$$\begin{aligned}\Phi(x_k + \alpha s_k, \rho_k) - \Phi(x_k, \rho_k) = & \ \alpha\langle s_k, g_k\rangle + \rho_k \left(\|c_k + \alpha A_k s_k\| - \|c_k\| \right) + O(\alpha^2) \\ = & \ \alpha\langle s_k, g_k\rangle + \rho_k \left(\|(1-\alpha) c_k\| - \|c_k\| \right) + O(\alpha^2) \\ = & \ \alpha \left(\langle s_k, g_k\rangle - \rho_k \|c_k\| \right) + O\left(\alpha^2\right).\end{aligned}$$

Combining this with (C.58), and once again using the positive definiteness of B_k, the Hölder inequality (that is that $\langle u, v\rangle \leq \|u\|\|v\|_D$ for any u, v) and the fact that $s_k \neq 0$ if x_k is not critical, yields

$$\begin{aligned}\Phi(x_k + \alpha s_k, \rho_k) - \Phi(x_k, \rho_k) = & \ -\alpha \left(\langle s_k, B_k s_k\rangle + \langle c_k, y_{k+1}\rangle + \rho_k \|c_k\| \right) + O(\alpha^2) \\ < & \ -\alpha \left(-\|c_k\|\|y_{k+1}\|_D + \rho_k \|c_k\| \right) + O(\alpha^2) \\ = & \ -\alpha \|c_k\| \left(\rho_k - \|y_{k+1}\|_D \right) + O(\alpha^2) < 0\end{aligned}$$

because of the required bound on ρ_k, for sufficiently small α. Hence sufficiently small steps along s_k from non-critical x_k reduce $\Phi(x, \rho_k)$.

Index