3 NUMERICAL OPTIMISATION

3.1 Introduction

3.1.1 Preliminaries

In general, optimisation problems can be stated as problems of minimisation of some function of the design parameters **x**, subjected to certain constraints, i.e.:

$$\begin{array}{lll} \textit{minimise} & f(\mathbf{x}), & \mathbf{x} \in \mathbb{R}^n \\ \textit{subject to} & c_i(\mathbf{x}) = 0, & i \in E \\ \textit{and} & c_j(\mathbf{x}) \geq 0, & j \in I\ , \end{array} \tag{3.1}$$

where $f(\mathbf{x})$ is the objective function and $c_i(\mathbf{x})$ and $c_j(\mathbf{x})$ are constraint functions[1]. Design parameters are also referred to as optimisation variables. The second line of (3.1) represents the equality constraints of the problem and the third line represents the inequality constraints. We have introduced two index sets, set E of the equality constraint indices and set I of the inequality constraint indices. The above problem is also referred to as the general nonlinear problem. Most of optimisation problems can be expressed in this form, eventually having multiple objective functions in the case of several conflicting design objectives.

Points $\mathbf{x}'$, which satisfy all constraints, are called feasible points and the set of all such points is called the feasible region. A point $\mathbf{x}^*$ is called a constrained local minimiser (or local solution of the above problem) if there exists some neighbourhood Ω of $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x}')$ for all feasible points $\mathbf{x}' \in \Omega, \mathbf{x}' \neq \mathbf{x}^*$. Such a point is called a strict local minimiser if the $<$ sign is applied in place of $\leq$; a

[1] Number of optimisation variables will be denoted by n throughout chapter 3.

slightly stronger definition of isolated local minimiser, which requires the minimiser to be the only local minimiser in some neighbourhood. Furthermore, **x*** is called the global solution or global constrained minimiser if $f\left(\mathbf{x}^*\right) \leq f\left(\mathbf{x}'\right)$ for all feasible points **x**'. This means that a global minimiser is the local solution with the least value of *f*.

Since the objective and constraint functions are in general nonlinear, the optimisation problem can have several constrained local minimisers $\mathbf{x}^*$. The goal of optimisation is of course to comply with the objective as much as possible, therefore the identification of the global solution is the most desirable. However, this problem is in general extremely difficult to handle. Actually there is no general way to prove that some point is a global minimiser. At best some algorithms are able to locate several local solutions and one can then take the best one of these. These methods are mostly based on some stochastic search strategy. Location of problem solutions is of a statistical nature, which inevitably leads to an enormous number of function evaluations needed to locate individual solutions with satisfactory accuracy and certainty. These methods are therefore usually not feasible for use with costly numerical simulations and are not included in the scope of this work. Currently the most popular types of algorithms for identifying multiple local solutions are the simulated annealing algorithms and genetic algorithms, briefly described in [9].

The optimisation problem can appear in several special forms dependent on whether the inequality or equality constraints are present or not, and whether the objective and constraint functions have some simple form (e.g. are linear or quadratic in the optimisation parameters). These special cases are interesting for mathematical treatment because it is usually possible to construct efficient solution algorithms that take advantage of the special structure.

In the cases related to this work, the objective and constraint functions are typically evaluated implicitly through a system response evaluated with complex numerical simulation. Here it can not be assumed that these functions will have any advantageous structure. At most there are cases with linear constraints or constraints that can be reduced to the form of simple bounds on variables, and in some cases it is possible to manage the problem without setting any constraints. Treatment of optimisation algorithms in this chapter will correspond to this fact. Some problems with special structure will however be considered since they appear as sub-problems in general algorithms. Example of this is the problem (3.1) with a quadratic objective function and linear constraint functions (the so called quadratic programming or QP problem), which often appears in algorithms for general constrained and unconstrained minimsation.

It proves that solution of the constrained problem is essentially more complex than solution of the unconstrained problem. Also theoretical treatment of the latter is in many aspects a natural extension of unconstrained minimisation, therefore the first part of this section is dedicated to the general unconstrained minimisation in multivariable space. Some attention is drawn to show parallels with solution of systems of nonlinear equations, which is the core problem in numerical simulations related to this work. The source of additional complexity that arises in practical unconstrained minimisation, as compared to the solution of nonlinear equations that appear in simulations, will be addressed. The aim of this section is to represent the theoretical background used in treatment of this complexity in order to assure satisfactory local and global convergence properties. Basic treatment of the one dimensional line search, a typical property of most practical algorithms, is also given in this context.

In the second part a more general constrained problem will be addressed. The additional mathematical background such as necessary and sufficient conditions will be given first. The two most commonly used approaches to constrained optimisation will then be described: sequential unconstrained minimisation and sequential quadratic programming.

The section is concluded with some practical considerations with regard to the present work. Some practical problems that can give rise to inadequacy of the described theory will be indicated. A problem strongly related to this work is optimisation in the presence of substantial amounts of numerical noise, which can cause serious difficulties to algorithms based on certain continuity assumptions regarding the objective and constraint functions.

3.1.2 Heuristic Minimisation Methods and Related Practical Problems

In the subsequent text the unconstrained problem is considered, namely

$$\textit{minimise} \qquad f(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^n \tag{3.2}$$

Throughout this chapter it is assumed that f is at least a $\mathbb{C}^2$ function, i.e. twice continuously differentiable with respect to $\mathbf{x}$. Every local minimum is a stationary point of f, i.e. a point with zero gradient[1]:

$$\nabla f\left(\mathbf{x}^*\right) = \mathbf{g}\left(\mathbf{x}^*\right) = \mathbf{g}^* = 0\,. \tag{3.3}$$

Minimisation can therefore be considered as a solution of the above equation, which is essentially a system of nonlinear equations for gradient components

$$g_i(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial x_i} = 0\,,\ \ i = 1, \ldots n\,. \tag{3.4}$$

This is essentially the same system that arises in finite element simulation[34] and can be solved by the standard Newton method, for which the iteration is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(\nabla \mathbf{g}^{(k)}\right)^{-1} \mathbf{x}^{(k)}\,. \tag{3.5}$$

The notation $\mathbf{g}^{(k)} = \mathbf{g}\left(\mathbf{x}^{(k)}\right)$ is adopted throughout this work.

The method is derived from the Taylor series[30],[32] for **g** about the current estimate $\mathbf{x}^{(k)}$:

$$\mathbf{g}\left(\mathbf{x}^{(k)} + \delta\right) = \mathbf{g}^{(k)} + \nabla \mathbf{g}^{(k)} \delta + O\left(\|\delta\|^2\right) \tag{3.6}$$

Considering this as the first order approximation for **g** and equating it to zero we obtain the expression for step δ which should bring the next estimate close to the solution of (3.4)[1]:

$$\nabla \mathbf{g}^{(k)} \delta = -\mathbf{g}^{(k)}\,.$$

By setting $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta$ we obtain the above Newton Iteration.

The Newton method is known to be rapidly convergent[2], but suffers for a lack of global convergence properties, i.e. the iteration converges to the solution only in some limited neighbourhood, but not from any starting point. This is the fundamental reason that it is usually not applicable to optimisation without modifications. The problem can usually be elegantly avoided in simulations, either because of some nice physical properties of the analysed system that guarantee global convergence, or by the ability of making the starting guess arbitrarily close to the equilibrium point where the equations are satisfied. This is, for example, exploited in the solution of path dependent problems where the starting guess of the current iterate is the equilibrium of the previous, and this can be set arbitrarily close to the solution because of the continuous nature of the governing equations. Global convergence can be ensured simply by cutting down the step size, if necessary.

In practice, this is usually not at all case in optimisation. The choice of a good starting point typically depends only on a subjective judgment where the solution should be, and the knowledge used for this is usually not sufficient to choose the

[1] Notation $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$, $f^{(k)} = f\left(\mathbf{x}^{(k)}\right)$, $\mathbf{g}^{(k)} = \mathbf{g}\left(\mathbf{x}^{(k)}\right)$, etc. will be generally adopted throughout this text.

starting point within the convergence radius of Newton's method, especially due to the complex non-linear behaviour of f and consequently **g**. Modifications to the method must therefore be made in order to induce global convergence[1], i.e. convergence from any starting guess.

One such modification arises from considering what properties the method must have in order to induce convergence to the solution. The solution $\mathbf{x}^*$ must be a limiting point of the sequence of iterations. This means that the distance between the iterates and the solution tends towards zero, i.e.

$$\lim_{k\to\infty} \left\| \mathbf{x}_k - \mathbf{x}^* \right\| = 0 . \tag{3.7}$$

This is satisfied if the above norm is monotonically decreasing and if the sequence has no accumulation point other than $\mathbf{x}^*$. When considering the minimisation problem and assuming that the problem has a unique solution, the requirements for a decreasing norm can be replaced (because of continuity of f) by the requirement that $f^{(k)}$ are monotonically decreasing. By such consideration, a basic property any minimisation algorithm should have, is the generation of descent iterates so that

$$f^{(k+1)} < f^{(k)} \quad \forall k . \tag{3.8}$$

This is closely related to the idea of line search, which is one of the elementary ideas in construction of minimisation algorithms. The idea is to minimise f along some straight line starting from the current iterate. Many algorithms are centered on this idea, trying to generate a sequence of directions along which line searches are performed, such that a substantial reduction of f is achieved in each line search and such that, in the limit, the rapid convergence properties of Newton's method are inherited.

An additional complication which limits the applicability of Newton's method is that the second derivatives of the objective function (i.e. first derivatives of its gradient) are required. These are not always directly available since double differentiation of numerical models is usually a much harder problem than single differentiation. Alternatively the derivatives can be obtained by straight numerical differentiation using small perturbation of parameters, but in many cases this is not applicable because numerical differentiation is very sensitive to errors in function evaluation[31],[33], and these can often not be avoided sufficiently when numerical models with many degrees of freedom are used. Furthermore, even if the Newton method converges, the limiting point is only guaranteed to be a stationary point of f,

[1] Herein the expression global convergence is used to denote convergence to a local solution from any given starting point. In some of the literature this expression is used to denote convergence to a global solution.

but this is not a sufficient condition for a local minimum, since it includes saddle points, which are stationary points but are not local minimisers.

The most simple algorithm that incorporates the idea of line search is sequential minimisation of the objective function in some fixed set of *n* independent directions in each iterate, most elementarily parallel to the coordinate axes. The requirement for *n* independent directions is obvious since otherwise the algorithm could not reach any point in $\mathbb{R}^n$. The method is called the alternating variables method and it seems to be adequate at a first glance, but turns out to be very inefficient and unreliable in practice. A simple illustration of the reasons for this is that the algorithm ignores the possibility of correlation between the variables. This causes the search parallel to the current search direction to destroy completely the property that the current point is the minimiser in previously used directions. This leads to oscillatory behaviour of the algorithm as illustrated in Figure 3.1.

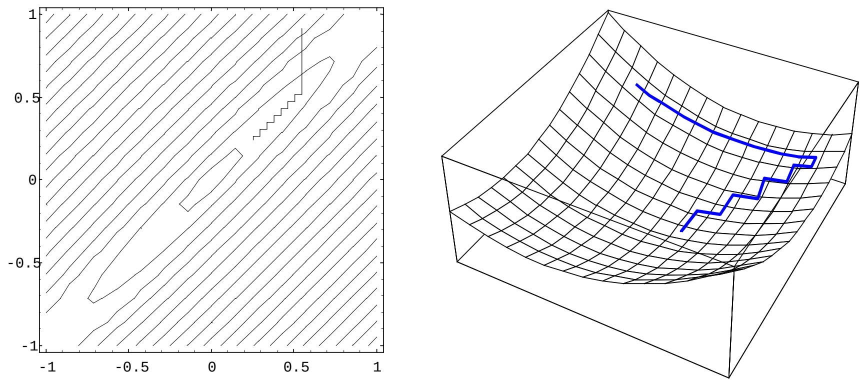

Figure 3.1: Oscillatory behaviour, which is likely to occur when using sequential minimisation in a fixed set of directions.

Another readily available algorithm is sequential minimisation along the current direction of the gradient of *f*. Again this seems to be a good choice, since the gradient is the direction of the steepest descent, i.e. the direction in which *f* decreases most rapidly in the vicinity of the starting point. With respect to this, the method is called the steepest descent method. In practice, however, the method suffers for similar problems to the alternating variables method, and the oscillating behaviour of this method is illustrated in Figure 3.2. The theoretical proof of convergence exists, but it can also be shown that locally the method can achieve an arbitrarily slow rate of linear convergence[1].

The above discussion clearly indicates the necessity for a more rigorous mathematical treatment of algorithms. Indeed the majority of the up-to-date algorithms have a solid mathematical background[1]-[7], [26] and partially the aim of this section is to point which are the most important features in the design of fast and reliable algorithms.

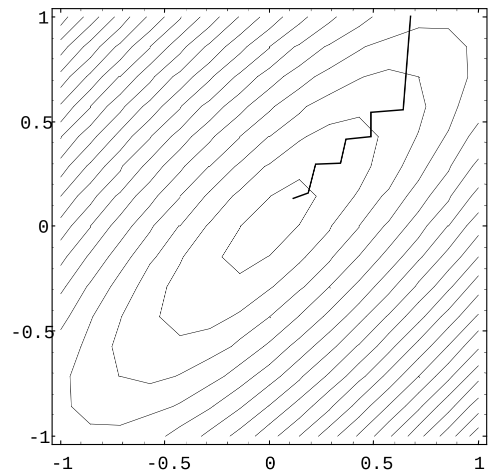

Figure 3.2: Oscillatory behaviour, which can occur when performing sequential line searches along the steepest descent directions.

3.2 Simplex Method

One minimisation method that does not belong within the context of the subsequent text is the simplex method[12], [26],[1]. It has been known since the early sixties and could be classed as another heuristic method since it is not based on a substantial theoretical background.

The simplex method neither uses line searches nor is based on minimisation of some simplified model of the objective function, and therefore belongs to the class of direct search methods. Because of this the method does not compare well with other described methods with respect to local convergence properties. On the other hand, for the same reason it has some other strong features. The method is relatively insensitive to numerical noise and does not depend on some other properties of the objective function (e.g. convexity) since no specific continuity or other assumptions are incorporated in its design. It merely requires the evaluation of function values. Its performance in practice can be as satisfactory as any other non-derivative method, especially when high accuracy of the solution is not required and the local convergence properties of more sophisticated methods do not play so important role. In many cases it does not make sense to require highly accurate solutions of optimisation problems, because the obtained results are inevitably inaccurate with respect to real system behaviour due to numerical modeling of the system (e.g. discretisation and round-off errors or inaccurate physical models). These are definitely good arguments for considering practical use of the method in spite of the lack of good local convergence results with respect to some other methods.

The simplex method is based on construction of an evolving pattern of *n+1* points in $\mathbb{R}^n$ (vertices of a simplex). The points are systematically moved according to some strategy such that they tend towards the function minimum. Different strategies give rise to different variants of the algorithm. The most commonly used is the Nelder-Mead algorithm described below. The algorithm begins by choice of *n+1* vertices of the initial simplex ($\mathbf{x}_1^{(1)}, \ldots, x_{n+1}^{(1)}$) so that it has non-zero volume. This means that all vectors connecting a chosen vertex to the reminding vertices must be linearly independent, e.g.

$$\exists \lambda_i \neq 0 \Rightarrow \sum_{i=1}^{n} \lambda_i \left(\mathbf{x}_{i+1}^{(1)} - \mathbf{x}_1^{(1)}\right) \neq 0 .$$

If we have chosen $\mathbf{x}_1^{(1)}$, we can for example obtain other vertices by moving, for some distance, along all coordinate directions. If it is possible to predict several points that should be good according to experience, it might be better to set vertices to these points, but the condition regarding independence must then be checked.

Once the initial simplex is constructed, the function is evaluated at its vertices. Then one or more points of the simplex are moved in each iteration, so that each subsequent simplex consists of a better set of points:

Algorithm 3.1: The Nelder-Mead simplex method.

After the initial simplex is chosen, function values in its vertices are evaluated: $f_i^{(1)} = f\left(\mathbf{x}_i^{(1)}\right), i = 1, \ldots, n+1$.

Iteration *k* is then as follows:

1. **Ordering step:** Simplex vertices are first reordered so that $f_1^{(k)} \leq f_2^{(k)} \leq \ldots \leq f_{n+1}^{(k)}$, where $f_i^{(k)} = f\left(\mathbf{x}_i^{(k)}\right)$.
2. **Reflection step:** The worst vertex is reflected over the centre point of the best *n* vertices ($\overline{\mathbf{x}}^{(k)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^{(k)}$), so that the reflected point $\mathbf{x}_r^{(k)}$ is

$$\mathbf{x}_r^{(k)} = \overline{\mathbf{x}}^{(k)} + \left(\overline{\mathbf{x}}^{(k)} - \mathbf{x}_{n+1}^{(k)}\right)$$

Evaluate $f_r^{(k)} = f\left(\mathbf{x}_r^{(k)}\right)$. If $f_1^{(k)} \le f_r^{(k)} < f_n^{(r)}$, accept the reflected point and go to 6.

3. **Expansion step:** If $f_r^{(k)} < f_1^{(k)}$, calculate the expansion

$$\mathbf{x}_e^{(k)} = \overline{\mathbf{x}}^{(k)} + 2\left(\mathbf{x}_r^{(k)} - \overline{\mathbf{x}}^{(k)}\right)$$

and evaluate $f_e^{(k)} = f\left(\mathbf{x}_e^{(k)}\right)$. If $f_e^{(k)} < f_r^{(k)}$, accept $\mathbf{x}_e^{(k)}$ and go to 6. Otherwise accept $\mathbf{x}_r^{(k)}$ and go to 6.

4. **Contraction step:** If $f_r^{(k)} \ge f_n^{(k)}$, perform contraction between $\overline{\mathbf{x}}^{(k)}$ and the better of $\mathbf{x}_{n+1}^{(k)}$ and $\mathbf{x}_r^{(k)}$. If $f_r^{(k)} < f_{n+1}^{(k)}$, set

$$\mathbf{x}_c^{(k)} = \overline{\mathbf{x}}^{(k)} + \frac{1}{2}\left(\mathbf{x}_r^{(k)} - \overline{\mathbf{x}}^{(k)}\right)$$

(this is called the outside contraction) and evaluate $f_c^{(k)} = f\left(\mathbf{x}_c^{(k)}\right)$. If $f_c^{(k)} \le f_r^{(k)}$, accept $\mathbf{x}_c^{(k)}$ and go to 6.
If in contrary $f_r^{(k)} \ge f_{n+1}^{(k)}$, set

$$\mathbf{x}_c^{(k)} = \overline{\mathbf{x}}^{(k)} - \frac{1}{2}\left(\overline{\mathbf{x}}^{(k)} - \mathbf{x}_{n+1}^{(k)}\right)$$

(inside contraction) and evaluate $f_c^{(k)}$. If $f_c^{(k)} < f_{n+1}^{(k)}$, accept $\mathbf{x}_c^{(k)}$ and go to 6.

5. **Shrink step:** Move all vertices except the best towards the best vertex, i.e.

$$\mathbf{v}_i^{(k)} = \mathbf{x}_1^{(k)} + \frac{1}{2}\left(\mathbf{x}_i^{(k)} - \mathbf{x}_1^{(k)}\right), i = 2, ..., n+1,$$

and evaluate $f_i^{'(k)} = f\left(\mathbf{v}_i^{(k)}\right), i = 2, ..., n+1$. Accept $\mathbf{v}_i^{(k)}$ as new vertices.

6. **Convergence check:** Check if the convergence criterion is satisfied. If so, terminate the algorithm, otherwise start the next iteration.

Figure 3.3 illustrates possible steps of the algorithm. A possible situation of two iterations when the algorithm is applied is shown in Figure 3.4. The steps allow the shape of the simplex to be changed in every iteration, so the simplex can adapt to the surface of *f*. Far from the minimum the expansion step allows the simplex to move rapidly in the descent direction. When the minimum is inside the simplex, contraction and shrink steps allow vertices to be moved closer to it.

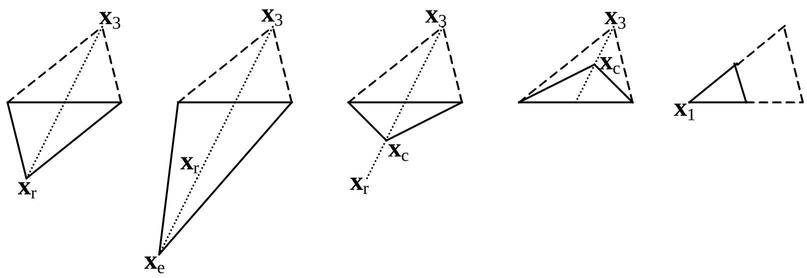

Figure 3.3: Possible steps of the simplex algorithm in two dimensions (from left to right): reflection, expansion, outside and inside contraction, and shrink.

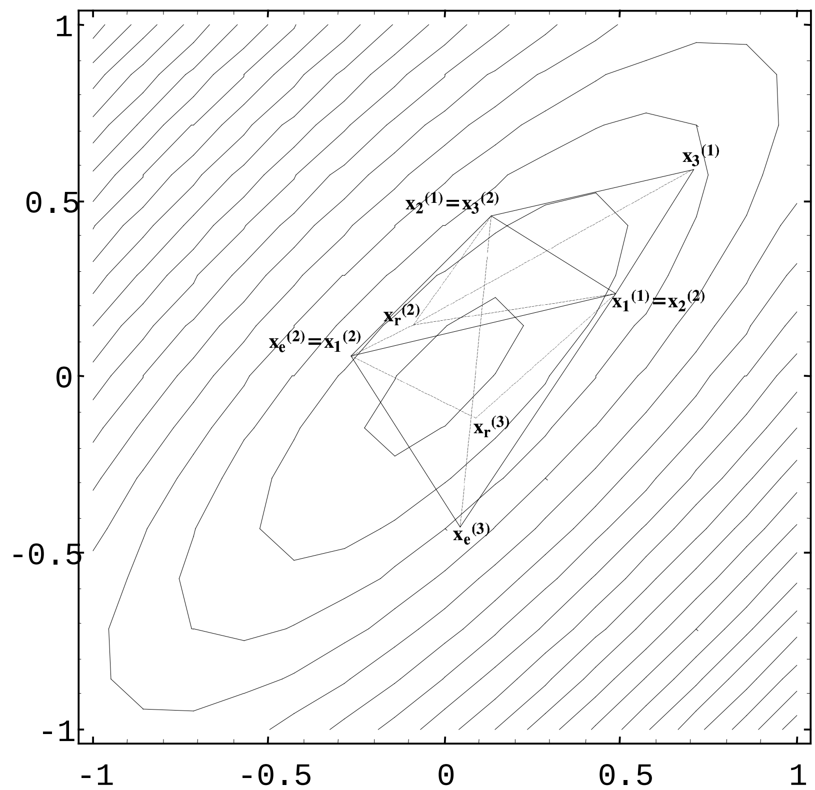

Figure 3.4: Example of evolution of the simplex.

There are basically two possibilities for the convergence criterion. Either that function values at vertices must become close enough or the simplex must becomes small enough. It is usually best to impose both criteria, because either of them alone can be misleading.

It must be mentioned that convergence to a local minimum has not been proved for the Nelder-Mead algorithm. Examples have been constructed for which the method does not converge[12]. However, the situations for which this was shown are quite special and unlikely to occur in practice. Another theoretical argument against the algorithm is that it can fail because the simplex collapses into a subspace, so that vectors connecting its vertices become nearly linearly dependent. Investigation of this phenomenon indicates that such behaviour is related to cases when the function to be minimised has highly elongated contours (i.e. ill conditioned Hessian). This is also a problematic situation for other algorithms.

The Nelder-Mead algorithm can be easily adapted for constrained optimisation. One possibility is to add a special penalty term to the objective function, e.g.

$$f'(\mathbf{x}) = f(\mathbf{x}) + f_{n+1}^{(1)} - \sum_{i \in I} c_i(\mathbf{x}) + \sum_{i \in I} \left| c_j(\mathbf{x}) \right|, \tag{3.9}$$

where $f_{n+1}^{(1)}$ is the highest value of f in the vertices of the initial simplex. Since subsequent iterates generate simplices with lower values of the function at vertices, the presence of this term guarantees that whenever a trial point in some iteration violates any constraints, its value is greater than the currently best vertex. The last two sums give a bias towards the feasible region when all vertices are infeasible. The derivative discontinuity of the terms with absolute value should not be problematic since the method is not based on any model, but merely on comparison of function values. A practical implementation is similar to the original algorithm. f is first evaluated at the vertices of the initial simplex and the highest value is stored. Then the additional terms in (3.9) are added to these values, and in subsequent iterates f is replaced by f'.

Another variant of the simplex method is the multidirectional search algorithm. Its iteration consists of similar steps to the Nelder-Mead algorithm, except that all vertices but the best one are involved in all operations. There is no shrink step and the contraction step is identical to the shrink step of the Nelder-Mead algorithm. Possible steps are shown in Figure 3.5. The convergence proof exists for this method[12], but in practice it performs much worse than the Nelder-Mead algorithm. This is due to the fact that more function evaluations are performed at each iteration and that the simplex can not be adapted to the local function properties as well as the former algorithm. The shape of the simplex can not change, i.e. angles between it edges remain constant (see Figure 3.5). The multidirectional search algorithm is better suited to parallel processing because *n* function evaluations can always be performed simultaneously.

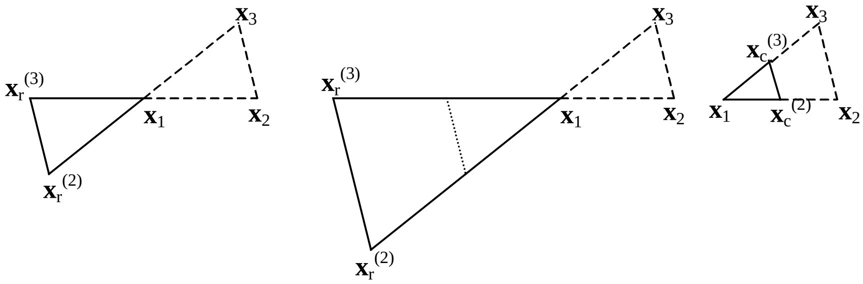

Figure 3.5: possible steps in the multidirectional search algorithm: reflection, expansion, and contraction.

3.3 Basic Mathematical Background

Construction of optimisation methods described further in this section is based on some model of the objective function and constraints. Such treatment of the problem arises to a large extent from the fact that locally every function can be developed into a Taylor series[30] about any point $x^{'}$:

$$f\left(x^{'}+h\right)=\sum_{n=0}^{\infty}\frac{h^n}{n!}f^{(n)}\left(x^{'}\right), \tag{3.10}$$

where $f^{(n)}(x)=\frac{\partial^n}{\partial x^n}f(x)$ and $n!=1\cdot 2\cdot 3\cdot ...\cdot n$. This expression itself does not have a significant practical value. A more important fact is that

$$\lim_{n\to\infty}R_n(h)=0 \tag{3.11}$$

and

$$\lim_{h \to 0} R_n(h) = 0\,, \tag{3.12}$$

where

$$R_n(h) = f\left(x^{'} + h\right) - S_n(h) \tag{3.13}$$

and

$$S_n(h) = \sum_{i=0}^{n} \frac{h^n}{n!} f^{(n)}\left(x^{'}\right). \tag{3.14}$$

This means that if we use only a few terms in the Taylor series, the error that we make tends to zero both when we increase the number of terms without limit for some fixed *h*, and when we take a fixed number of terms and decrease the step *h* towards zero. This follows from the result[30]

$$R_n(h) = \frac{h^{n+1}}{(n+1)!} f^{(n+1)}\left(x^{'} + \theta h\right), 0 < \theta < 1\,. \tag{3.15}$$

The above equation also holds if function *f* is only $\mathbb{C}^{n+1}$. This means that every sufficiently smooth function can be locally approximated by a simple polynomial function, which is sometimes more convenient for theoretical treatment than the original function.

A similar development is possible for a function of *n* variables[30]:

$$\begin{aligned} & f\left(x_1^{'} + h_i, x_2^{'} + h_2, ..., x_n^{'} + h_n\right) = f\left(x_1^{'}, x_2^{'}, ..., x_n^{'}\right) + \\ & \sum_{i=1}^{m} \frac{1}{i!} \left(h_1 \frac{\partial}{\partial x_1} + h_2 \frac{\partial}{\partial x_2} + ... + h_n \frac{\partial}{\partial x_n} \right)^i f\left(x_1, x_2, ..., x_n\right) + \,, \\ & R_m\left(h_i, h_2, ..., h_n\right) \end{aligned} \tag{3.16}$$

where

$$\begin{aligned} & R_m\left(h_1, ..., h_n\right) = \frac{1}{(n+1)!} \left(h_1 \frac{\partial}{\partial x_1} + ... + h_n \frac{\partial}{\partial x_n} \right)^{m+1} \quad . \\ & f\left(x_1 + \theta_1 h_1, ..., x_n + \theta_n h_n\right), \quad 0 < \theta_i < 1,\ i = 1, ..., n \end{aligned} \tag{3.17}$$

In view of the beginning of this discussion, we can consider numerical optimisation as the estimation of a good approximation of the optimisation problem solution on the basis of limited information about the function, usually objective and constraint function values and their derivatives in some discrete set of points. The goal is to achieve satisfactory estimation with as little function and derivative evaluations as possible. Now we can use the fact that general functions can be locally approximated by simpler functions. Besides, functions of simple and known form (e.g. linear or quadratic) are completely described by a finite number of parameters. If we know these parameters, we know (in principle) all about the function, including minimising points.

There exists a clear correspondence between the above considerations and the design of optimisation algorithms. One thing to look at when constructing algorithms is how they perform on simple model functions, and proofs of local convergence properties based to a large extent on properties of the algorithms when applied to such functions[1]-[7].

Heuristically this can be explained by considering a construction of a minimisation algorithm in the following way. Use function values and derivatives in a set of points to build a simple approximation model (e.g. quadratic), which will be updated when new information is obtained. Consider applying an effective minimisation technique adequate for the model function. Since the model approximates the function locally, some information obtained in this way should be applicable to making decision where to set the next iterate when minimising the original function. In the limit, when the iterates approach the minimum, the model function should be increasingly better approximation and minima of the successively built models should be good guesses for the subsequent iterates.

In fact many algorithms perform in a similar manner. The difference is usually that models are not built directly, but the iterates are rather constructed in such a way that the algorithm has certain properties when applied to simple functions, e.g. termination in a finite number of steps. This ensures good local convergence properties. In addition some strategy must be incorporated which ensures global convergence properties of the algorithm. The remainder of this section will consider some mathematical concepts related to this. First, some basic notions will be introduced, and then some important algorithmic properties will be discussed.

3.3.1 Basic Notions

Quadratic model functions are the most important in the study of unconstrained minimisation. This is because the Taylor series up to quadratic terms is the simplest Taylor approximation that can have an unconstrained local minimum.

Keeping the terms up to the second order in (3.16) gives the following expression for a second order Taylor approximation:

$$f\left(\mathbf{x}^{'}+\mathbf{h}\right)\approx f\left(\mathbf{x}^{'}\right)+\mathbf{h}^{T}\nabla f\left(\mathbf{x}^{'}\right)+\frac{1}{2}\mathbf{h}^{T}\left[\nabla^{2} f\left(\mathbf{x}^{'}\right)\right]\mathbf{h}, \tag{3.18}$$

where

$$\nabla f(\mathbf{x})=\mathbf{g}(\mathbf{x})=\left[\frac{\partial f}{\partial x_1},\frac{\partial f}{\partial x_2},\ldots,\frac{\partial f}{\partial x_n}\right]_{\mathbf{x}}^{T}$$

is the function gradient and

$$\nabla^{2} f(\mathbf{x})=\mathbf{G}(\mathbf{x})=\left(\nabla\nabla^{T}\right)f(\mathbf{x})$$

is the Hessian matrix[1] of the function, i.e. matrix of function second derivatives,

$$\left[\nabla^{2} f(\mathbf{x})\right]_{ij}=\mathbf{G}_{ij}(\mathbf{x})=\frac{\partial^{2} f}{\partial x_i \partial x_j}(\mathbf{x}). \tag{3.19}$$

Notation $\mathbf{g}(\mathbf{x})=\nabla f(\mathbf{x})$ and $\mathbf{G}(\mathbf{x})=\nabla^{2} f(\mathbf{x})$ will be used throughout this text.

The idea of a line in $\mathbb{R}^n$ is important. This is a set of points

$$\mathbf{x}=\mathbf{x}(\alpha)=\mathbf{x}^{'}+\alpha\mathbf{s}, \tag{3.20}$$

where $\alpha\in\mathbb{R}$ is a scalar parameter, $\mathbf{x}^{'}$ is any point on the line and **s** is the direction of the line. **s** can be normalised, e.g. with respect to the Euclidian norm, i.e. $\sum_{i=1}^{n} s_i^2=1$.

It is often useful to study how a function defined in $\mathbb{R}^n$ behaves on a line. For this purpose, we can write

[1] In standard notation Operator $\nabla^2=\Delta=\nabla^T\nabla=\sum_{i=1}^{n}\frac{\partial^2}{\partial x_i^{\,2}}$ is the Laplace operator. However, in most optimisation literature this notation is used for the Hessian operator, and so is also used in this text.

$$f(\alpha) = f(\mathbf{x}(\alpha)) = f(\mathbf{x}' + \alpha\mathbf{s}). \tag{3.21}$$

From this expression we can derive direction derivative of *f*, i.e. derivative of the function along the line:

$$\frac{df(\alpha)}{d\alpha} = \sum_i \frac{dx_i}{d\alpha}\frac{\partial f}{\partial x_i} = \sum_i s_i \frac{\partial f}{\partial x_i} = (\nabla f(\mathbf{x}(\alpha)))^T \mathbf{s}\,.$$

This can be written as

$$\frac{df}{d\alpha} = \frac{df}{d\mathbf{s}} = \nabla f^T \mathbf{s}\,. \tag{3.22}$$

In a similar way the curvature along the line is obtained:

$$\frac{d^2 f(\alpha)}{d\alpha^2} = \frac{d}{d\alpha}\frac{df}{d\alpha} = \frac{d}{d\alpha}\sum_{i=1}^{n} s_i \frac{\partial f}{\partial x_i} =$$
$$\sum_{i=1}^{n} s_i \sum_{j=1}^{n} \frac{dx_j}{d\alpha}\frac{\partial^2 f}{\partial x_i \partial x_j} = \sum_{i=1}^{n}\sum_{j=1}^{n} s_i s_j \frac{\partial^2 f}{\partial x_i \partial x_j}$$

and so

$$\frac{d^2 f}{d\alpha^2} = \frac{d^2 f}{d\mathbf{s}^2} = \mathbf{s}^T \left(\nabla^2 f\right)\mathbf{s}\,. \tag{3.23}$$

A general quadratic function can be written in the form

$$q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{G}\mathbf{x} + \mathbf{b}^T \mathbf{x} + c\,, \tag{3.24}$$

where **G** is a symmetric constant matrix, $\mathbf{b}^T$ a constant vector and *c* a constant scalar. The gradient of this function is

$$\nabla q(\mathbf{x}) = \mathbf{G}\mathbf{x} + \mathbf{b} \tag{3.25}$$

and the Hessian matrix is

$$\nabla^2 q(\mathbf{x}) = \mathbf{G}\,, \tag{3.26}$$

where the rule for gradient of a vector product

$$\nabla\left(\mathbf{u}^T\mathbf{v}\right)=\left(\nabla\mathbf{u}^T\right)\mathbf{v}+\left(\nabla\mathbf{v}^T\right)\mathbf{u};\;\; \mathbf{u}=\mathbf{u}(\mathbf{x}),\, \mathbf{v}=\mathbf{v}(\mathbf{x})$$

was applied.

We see that a quadratic function has a constant Hessian and its gradient is an affine function of **x**. As a consequence, for any two points the following equation relating the gradient in these points is valid:

$$\nabla q\left(\mathbf{x}''\right)-\nabla q\left(\mathbf{x}'\right)=\mathbf{G}\left(\mathbf{x}''-\mathbf{x}'\right). \tag{3.27}$$

If **G** is nonsingular, a quadratic function has a unique stationary point ($\nabla q\left(\mathbf{x}'\right)=0$):

$$\mathbf{x}'=-\mathbf{G}^{-1}\mathbf{b}\,, \tag{3.28}$$

which is also a minimiser if **G** is positive definite (see section 3.3.2). Taylor development about the stationary point gives another form for a quadratic function

$$q(\mathbf{x})=\frac{1}{2}\left(\mathbf{x}-\mathbf{x}'\right)^T\mathbf{G}\left(\mathbf{x}-\mathbf{x}'\right)+c'\,, \tag{3.29}$$

where $c'=c-\frac{1}{2}\mathbf{x}'^T\mathbf{G}\mathbf{x}'$.

In this text a term linear function[1] will be used for functions of the form

$$l(\mathbf{x})=\mathbf{a}^T\mathbf{x}+b\,, \tag{3.30}$$

where $\mathbf{a}^T$ is a constant vector and *b* a constant scalar. Such functions have a constant gradient

[1] Mathematically this is an affine function. Linear functions are those[30] for which $f(a\mathbf{x}+b\mathbf{y})=af(\mathbf{x})+bf(\mathbf{y})$ for arbitrary **x** and **y** in the definition domain and for arbitrary constants *a* and *b*. Affine functions are those for which $f(\mathbf{x})-c$ is a linear function, where *c* is some constant. However, in the optimisation literature affine functions are often referred to simply as linear and this is also adopted in this text.

$$\nabla l(\mathbf{x}) = \mathbf{a} \tag{3.31}$$

and zero Hessian

$$\nabla^2 l(\mathbf{x}) = 0\,. \tag{3.32}$$

3.3.2 Conditions for Unconstrained Local Minima

Consider first a line through some point $\mathbf{x}^*$, i.e. $\mathbf{x}(\alpha) = \mathbf{x}^* + \alpha\mathbf{s}$. Let us define a scalar function of parameter α using values of function f on this line as $f(\alpha) = f(\mathbf{x}(\alpha))$. If $\mathbf{x}^*$ is a local minimiser of $f(\mathbf{x})$, then 0 is clearly a local minimiser of $f(\alpha)$. From the Taylor expansion for a function of one variable about 0 then it follows[1] that f has zero slope and non-negative curvature at $\alpha = 0$. This must be true for any line through $\mathbf{x}^*$, and therefore for any $\mathbf{s}$. From (3.22) and (3.23) it then follows

$$\mathbf{g}^* = 0 \tag{3.33}$$

and

$$\mathbf{s}^T \mathbf{G}^* \mathbf{s} \geq 0 \quad \forall \mathbf{s}\,, \tag{3.34}$$

where the following notation is used: $f^* = f(\mathbf{x}^*)$, $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$, $\mathbf{g}^* = \mathbf{g}(\mathbf{x}^*)$, $\mathbf{G}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$, and $\mathbf{G}^* = \mathbf{G}(\mathbf{x}^*)$. This notation will be used through this text, and similarly $f(\mathbf{x}^{(k)}) = f^{(k)}$, etc.

Since (3.33) and (3.34) are implied by assumption that $\mathbf{x}^*$ is a local minimiser of *f*, these are necessary conditions for $\mathbf{x}^*$ being a local minimiser. (3.33) is referred to a first order necessary condition and (3.34) as a second order necessary condition. This condition states that the Hessian matrix is positive semi-definite in a local minimum.

The above necessary conditions are not at the same time sufficient, i.e. these conditions do not imply $\mathbf{x}^*$ to be a local minimiser. Sufficient conditions can be stated in the following way[1]:

Theorem 3.1:

Sufficient conditions for a strict and isolated local minimiser $\boldsymbol{x}^*$ of f are that f has a zero gradient and a positive definite Hessian matrix in $\boldsymbol{x}^*$:

$$\mathbf{g}^* = 0 \tag{3.35}$$

and

$$\mathbf{s}^T \mathbf{G}^* \mathbf{s} > 0 \quad \forall \mathbf{s} \neq 0 \tag{3.36}$$

There are various ways how to check the condition (3.36). The most important for practical purposes are that[27],[29] **G** is positive definite, the Choleski factors of the LL^T decomposition exist and all diagonal elements l_{ii} are greater than zero, and the same applies for diagonal elements d_{ii} of the LDL^T decomposition. This can be readily verified on those algorithms which solve a system of equation with the system matrix **G** in each iteration, since one of these decompositions is usually applied to solve the system.

Some algorithms do not evaluate the Hessian matrix. These can not verify the sufficient conditions directly. Sometimes these algorithms check only the first order condition or some condition based on the progress during the last few iterations. It can usually be proved that under certain assumptions iterates still converges to a local minimum. Algorithms should definitely have the possibility of termination in a stationary point, which is not a minimum (usually in a saddle point with indefinite Hessian matrix). Some algorithms generate subsequent approximations of the Hessian matrix, which converge to the Hessian in the limit when iterates approach a stationary point. The condition can then be checked indirectly on the approximate Hessian. More details concerning this will be outlined in the description of individual algorithms.

3.3.3 Desirable Properties of Algorithms

A desired behaviour of an optimisation algorithm is that iterates move steadily towards the neighbourhood of a local minimser, then converge rapidly to this point and finally that it identifies when the minimiser is determined with a satisfactory accuracy and terminates.

Optimisation algorithms are usually based on some model and on some prototype algorithm. A model is some approximation (not necessarily explicit) of the objective function, which enables a prediction of a local minimiser to be made.

A prototype algorithm refers to the broad strategy of the algorithm. Two basic types are the restricted step approach and the line search approach, described in detail in the subsequent sections. There it will be also pointed out that the ideas of prototype algorithms are usually closely associated with global convergence.

Local convergence properties of an algorithm describe its performance in the neighbourhood of a minimum. If we define the error of the *k*-th iterate

$$\mathbf{h}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^{*}, \tag{3.37}$$

it may be possible to state some limit results for $\mathbf{h}^{(k)}$. An algorithm is of course convergent if $\mathbf{h}^{(k)} \to 0$. If a limit

$$\lim_{k\to\infty} \frac{\left\|\mathbf{h}^{(k+1)}\right\|}{\left\|\mathbf{h}^{(k)}\right\|^{p}} = a \tag{3.38}$$

exists where $a > 0$ is some constant, then we say that the order of convergence is *p*. This definition can also be stated in terms of bounds if the limit does not exist: the order of convergence is *p* if

$$\frac{\left\|\mathbf{h}^{(k+1)}\right\|}{\left\|\mathbf{h}^{(k)}\right\|^{p}} \le a \tag{3.39}$$

for some constant $a > 0$ and for each *k* greater than some k_{lim}. An important cases are linear or first order convergence

$$\frac{\left\|\mathbf{h}^{(k+1)}\right\|}{\left\|\mathbf{h}^{(k)}\right\|} \le a \tag{3.40}$$

and quadratic or second order convergence

$$\frac{\left\|\mathbf{h}^{(k+1)}\right\|}{\left\|\mathbf{h}^{(k)}\right\|^{2}} \le a\,. \tag{3.41}$$

The constant *a* is called the rate of convergence and must be less than 1 for linear convergence. Linear convergence is only acceptable if the rate of convergence is small. If the order and rate are 1, the convergence is sublinear (slower than all linear convergence). This would be the case if $\left\|\mathbf{h}^{k}\right\| = 1/k$.

When the order is 1, but the rate constant is 0, the convergence is superlinear (faster than all linear convergence), i.e.

$$\lim_{k\to\infty} \frac{\left\|\mathbf{h}^{(k+1)}\right\|}{\left\|\mathbf{h}^{(k)}\right\|} = 0\,. \tag{3.42}$$

Successful methods for unconstrained minimisation converge superlinearly.

Many methods for unconstrained minimisation are derived from quadratic models. They are designed so that they work well or exactly on a quadratic function. This is partially associated with the discussion of section 3.3.1: since a general function is well approximated by a quadratic function, the quadratic model should imply good local convergence properties. Because the Taylor series about an arbitrary point taken to quadratic terms will agree to a given accuracy with the original function on a greater neighbourhood than the series taken to linear terms, it is preferable to use quadratic information even remote from the minimum.

The quadratic model is most directly used in the Newton method (3.5), which requires the second derivatives. A similar quadratic model is used in restricted step methods. When second derivatives are not available, they can be estimated in various ways. Such quadratic models are used in the quasi-Newton methods.

Newton-like methods (Newton or quasi-Newton) use the Hessian matrix or its approximation in Newton's iteration (3.5). A motivation for this lies in the Dennis-Moré theorem, which states that superlinear convergence can be obtained if and only if the step is asymptotically equal to that of the Newton-Raphson method[1].

The quadratic model is also used by the conjugate direction methods, but in a less direct way. Nonzero vectors $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, ..., \mathbf{s}^{(n)}$ are conjugate with respect to a positive definite matrix **G**, when

$$\mathbf{s}^{(i)T}\mathbf{G}s^{(j)} = 0 \;\; \forall\, i \neq j\,. \tag{3.43}$$

Optimisation methods, which generate such directions when applied to a quadratic function with Hessian **G**, are called conjugate direction methods. Such methods have the following important property[1]:

Theorem 3.2:

A conjugate direction method terminates for a quadratic function in at most n exact line searches, and each $\mathbf{x}^{(k)}$ is a minimiser of that function in the set

$$\left\{ \mathbf{x};\, \mathbf{x} = \mathbf{x}^{(1)} + \sum_{j=1}^{k} \alpha_j \mathbf{s}^{(j)} \;,\, \alpha_j \in \mathbb{R} \right\} \tag{3.44}$$

The above theorem states that conjugate direction methods have the property of quadratic termination, i.e. they can locate the minimising point of a quadratic function in a known finite number of steps. Many good minimisation algorithms can generate the set of conjugate directions, although it is not possible to state that superlinear convergence implies quadratic termination or vice versa. For example, some successful superlinearly convergent Newton-like methods do not possess this property.

It is useful to further develop the idea of conjugacy in order to gain a better insight in what it implies. We can easily see that $\mathbf{s}^{(i)}$ are linearly independent. If for example $\mathbf{s}^{(j)}$ was a linear combination of some other vectors $\mathbf{s}^{(k)}$, e.g.

$$\mathbf{s}^{(j)} = \sum_{k \neq j} \beta_k \mathbf{s}^{(k)} ,$$

multiplying this with $\mathbf{s}^{(j)T}\mathbf{G}$ would give

$$\mathbf{s}^{(j)T}\mathbf{G}\mathbf{s}^{(j)} = 0 ,$$

which contradicts the positive definiteness of **G**.

We can use vectors $\mathbf{s}^{(j)}$ as basis vectors and write any point as

$$\mathbf{x} = \mathbf{x}^{(1)} + \sum_{i=1}^{n} \alpha_i \mathbf{s}^{(i)} . \tag{3.45}$$

Taking into account this equation,(3.29) and conjugacy, the quadratic function from the theorem can be written as[1]

$$q(\alpha) = \frac{1}{2}\left(\mathbf{x} - \mathbf{x}^*\right)^T \mathbf{G}\left(\mathbf{x} - \mathbf{x}^*\right) = \frac{1}{2}\left(\alpha - \alpha^*\right)\mathbf{S}^T \mathbf{G}\mathbf{S}\left(\alpha - \alpha^*\right). \tag{3.46}$$

We have ignored a constant term in (3.29), which has no influence on further discussion, and written the minimiser $\mathbf{x}^*$ of q as

$$\mathbf{x}^{(*)} = \mathbf{x}^{(1)} + \sum \alpha_i^* \mathbf{s}^{(i)} ,$$

[1] Notation $\alpha = \left[\alpha_1, \alpha_2, ..., \alpha_n\right]^T$ is used. Vectors denoted by Greek letters are not typed in bold, but it should be clear from the context when some quantity is vector and when scalar.

and **S** is a matrix whose columns are vectors $\mathbf{s}^{(i)}$. Since $\mathbf{s}^{(i)}$ are conjugate with respect to **G**, the product $\mathbf{S}^{\mathrm{T}}\mathbf{G}\mathbf{S}$ is a diagonal matrix with diagonal elements d_i, say, and therefore

$$q(\alpha)=\frac{1}{2}\sum_{i=1}^{n}\left(\alpha_i-\alpha_i^*\right)^2 d_i\,. \tag{3.47}$$

We see that conjugacy implies a coordinate transformation from **x**-space to α-space in which **G** is diagonal. Variables in the new system are decoupled from the point of view that $q(\alpha)$ can be minimised by applying successive minimisations in coordinate directions, which results in a minimiser α^* corresponding to $\mathbf{x}^*$ in the **x** space. A conjugate direction method therefore corresponds to the alternating variable method applied in the new coordinate system. Enforcing conjugacy overcomes the basic problem associated with the alternating variable method, i.e. the fact that minimisation along one coordinate direction usually spoils earlier minimisations in other directions, which is the reason for oscillating behaviour of the method shown in Figure 3.1. Since a similar problem is associated with the steepest descent method, conjugacy can be successfully combined with derivative methods.

A side observation is that eigenvectors of **G** are orthogonal vectors conjugate to **G**. A quadratic function is therefore minimised by exact minimisation along all eigenvectors of its Hessian. Construction of the conjugate direction methods will show that there is no need to know eigenvectors of **G** in order to take advantage of conjugacy, but it is possible to construct conjugate directions starting with an arbitrary direction.

Another important issue in optimisation algorithms is when to terminate the algorithm. Since we can not check directly how close to the minimiser the current iterate is, the test can be based on conditions for a local minimum, for example

$$\left\|\mathbf{g}^{(k)}\right\|\le\varepsilon\,, \tag{3.48}$$

where ε is some tolerance. Sometimes it is not easy to decide what magnitude to choose for ε, since a good decision would require some clue about the curvature in the minimum. The above test is also dependent on the scaling of variables. Another difficulty is that it can terminate in a stationary point that is not a minimum. When second derivative information is available, it should be used to exclude this possibility.

When the algorithm converges rapidly, tests based on differences between iterates can be used, e.g.

$$\left|x_i^{(k)} - x_i^{(k+1)}\right| \le \varepsilon_i \ \forall i \tag{3.49}$$

or

$$f^{(k)} - f^{(k+1)} \le \varepsilon . \tag{3.50}$$

These tests rely on a prediction how much at most *f* can be further reduced or **x** approached to the minimum.

The test

$$\frac{1}{2}\mathbf{g}^{(k)T}\mathbf{H}^{k}\mathbf{g}^{(k)}, \tag{3.51}$$

where **H** is the inverse Hessian or its approximation, is also based on predicted change of *f*.

Finally, the possibility of termination when the number of iterations exceeds some user supplied limit is a useful property of every algorithm. Even when good local convergence results exist for a specific algorithm, this is not necessarily a guarantee for good performance in practice. Function evaluation is always subjected to numerical errors and this can especially affect algorithmic performance near the solution where local convergence properties should take effect.

3.4 Line Search Subproblem

3.4.1 Features Relevant for Minimisation Algorithms

The line search prototype algorithm sequentially minimises the objective function along straight lines. The structure of the *k*-th iteration is the following:

Algorithm 3.2: Iteration of a line search prototype algorithm.

1. Determine a search direction $\mathbf{s}^{(k)}$ according to some model.
2. Find $\alpha^{(k)}$, which minimises $f\left(\mathbf{x}^{(k)} + \alpha\,\mathbf{s}^{(k)}\right)$ and set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{s}^{(k)}$.

Finding a minimum of *f* on a line is referred to as the line search subproblem.

In the minimum, slope $df/d\alpha$ must be zero, which from (3.22) gives

$$\nabla f^{(k+1)T}\mathbf{s}^{(k)} = 0. \tag{3.52}$$

If $\mathbf{s}^{(k)}$ satisfies the descent property

$$\mathbf{s}^{(k)T}\mathbf{g}^{(k)} < 0, \tag{3.53}$$

then the function can be reduced in the line search for some $\alpha^{(k)} > 0$ unless $\mathbf{x}^{(k)}$ is a stationary point. A line search method in which search directions satisfy the descent property is called the descent method.

The descent property is closely associated with global convergence and by suitable choice of a line search condition it is possible to incorporate it within a global convergence proof. Merely requiring that *f* is decreased in each iteration certainly does not ensure global convergence. On the other hand, expensive high accuracy line searches do not make sense, especially when the algorithm is far from the solution. Therefore conditions for line search termination must be defined so that they allow low accuracy line searches, but still enforce global convergence.

Let us write $f(\alpha) = f\left(\mathbf{x}^{(k)} + \alpha\,\mathbf{s}^{(k)}\right)$ and let $\overline{\alpha}^{(k)}$ denote the least positive α for which $f(\alpha) = f(0)$ (Figure 3.6). Negligible reductions can occur if we allow the line search to be terminated in points close to 0 or $\overline{\alpha}^{(k)}$. Line search conditions must exclude such points, impose significant reductions of *f*, guarantee that acceptable points always exist and can be determined in a finite number of steps, and should not exclude the minimising point a^* when $f(\alpha)$ is quadratic with positive curvature.

These requirements are satisfied by the Goldstein conditions

$$f(\alpha) \le f(0) + \alpha\rho\, f'(0) \tag{3.54}$$

and

$$f(\alpha) \ge f(0) + \alpha(1-\rho) f'(0), \tag{3.55}$$

where $\rho \in \left(0, \frac{1}{2}\right)$ is some fixed parameter. (3.54) implies

$$f^{(k)} - f^{(k+1)} \geq -\rho \mathbf{g}^{(k)T} \delta^{(k)}, \tag{3.56}$$

where $\delta^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$. The condition $\rho < 0.5$ ensures that when $f(\alpha)$ is quadratic, the minimiser is an acceptable point, but this is not true for a general function (Figure 3.6 also shows the case where the minimiser is not an acceptable point). This deficiency is dismissed with the Wolfe-Powell conditions

$$f(\alpha) \leq f(0) + \alpha\rho\, f'(0) \tag{3.57}$$

and

$$f'(\alpha) \geq \sigma\, f'(0), \tag{3.58}$$

where $\rho \in \left(0, \frac{1}{2}\right)$ and $\sigma \in (\rho, 1)$. This implies

$$\mathbf{g}^{(k+1)T} \delta^{(k)} \geq \sigma\, \mathbf{g}^{(k)T} \delta^{(k)}. \tag{3.59}$$

Let $\hat{\alpha} > 0$ be the least positive value for which the graph $f(\alpha)$ intersects the line $f(0) + \alpha\,\rho\, f'(0)$ (point *b* in Figure 3.6). If such a point exists, then an interval of acceptable points for the Wolfe-Powell conditions exists in $(0, \hat{\alpha})$.

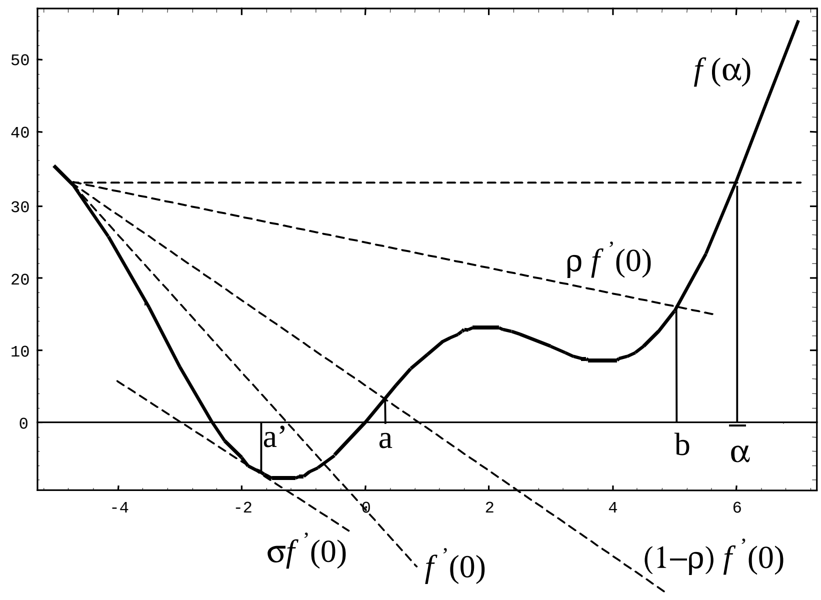

Figure 3.6: Line search conditions. [a,b] is the interval of acceptable points for Goldstein conditions, while [a’,b] is an interval of acceptable points for the Wolfe-Powell conditions. Slopes of auxiliary lines are denoted in the figure.

A two-sided test on the slope of f can also be used, i.e.

$$|f'(\alpha)| \leq -\sigma f'(0) \tag{3.60}$$

together with (3.57). An interval of acceptable points exists in $(0,\hat{\alpha})$ for this test, too, if $\hat{\alpha}$ exists.

To ensure global convergence, line searches must generate sufficient reduction of f in each iteration since otherwise non-minimising accumulation points of the sequence of iterates can exist. Fulfillment of this requirement depends on the applied line search criterion, but also on the line search directions. If these become orthogonal to the gradient, than no reduction of f can be made. It is advantageous to introduce some criterion to bound directions away from orthogonality to the gradient direction.

The angle criterion is defined as

$$\theta^{(k)} \leq \frac{\pi}{2} - \mu \quad \forall k\,, \tag{3.61}$$

where $\frac{\pi}{2} > \mu > 0$ is a fixed constant and $\theta^{(k)}$ is the angle between the gradient of f and the search direction, i.e.

$$\theta^{(k)} = \cos^{-1}\left(\frac{\left|\mathbf{g}^{(k)T}\mathbf{s}^{(k)}\right|}{\left\|\mathbf{g}^{(k)}\right\|_2 \left\|\mathbf{s}^{(k)}\right\|_2} \right). \tag{3.62}$$

The following global convergence theorem then holds[1]:

Theorem 3.3:

For a descent method with Wolfe-Powell line search conditions, if ∇f is uniformly continuous on the level set $\{\mathbf{x};\ f(\mathbf{x}) < f^{(1)}\}$ and if the angle criterion (3.61) holds, then either $f^{(k)} \to -\infty$ or $\mathbf{g}^{(k)} \to 0$.

Considering practical algorithms, the steepest descent method satisfies the angle criterion. Newton-like algorithms (section 3.5) define the search direction as

$$\mathbf{s}^{(k)} = -\mathbf{H}^{(k)}\mathbf{g}^{(k)}. \tag{3.63}$$

If $\mathbf{H}^{(k)}$ is positive definite, then $\mathbf{s}^{(k)}$ is a descent direction. In this case a sufficient condition is that the spectral condition number $\kappa^{(k)}$ of $\mathbf{H}^{(k)}$ is bounded above for every *k*. The spectral condition number of a matrix is the ratio between its largest and smallest eigenvalues (λ_1/λ_n). The relations[29],[33] $\|\mathbf{Hg}\|_2 \le \lambda_1 \|\mathbf{g}\|_2$ and $\mathbf{g}^T\mathbf{HG} \ge \lambda_n \mathbf{g}^T\mathbf{g}$ hold for any matrix **H** and vector **g**, and this implies an estimation

$$\theta^{(k)} \le \frac{\pi}{2} - \frac{1}{\kappa^{(k)}},$$

which implies the above statement.

A much weaker criterion can be used in place of the angle criterion in Theorem 3.3, namely

$$\sum_k \cos^2 \theta^{(k)} = \infty . \tag{3.64}$$

in this case $\liminf \left\|\mathbf{g}^{(k)}\right\| = 0$, which means that $\mathbf{g}^{(k)} \to 0$ on a subsequence[1].

3.4.2 Derivative Based Line Search Algorithms

Line search algorithms[1],[4] ,[13] consist of two parts. The first one is the bracketing stage, which finds a bracket, that is an interval known to contain acceptable points. The second part is the sectioning stage, in which a sequence of brackets whose length tends to zero is generated. It is advantageous to use some interpolation of $f(\alpha)$ in this stage in order to find an acceptable point which is close to the minimiser.

If *f* is not bounded below, it can happen that an interval of acceptable points does not exist. It is therefore advisable to supply a lower bound ($\bar{f}$, say) so that all points for which $f(a) \le \bar{f}$ are considered acceptable. The line search can then be restricted to the interval $(0, \mu]$, where μ is the point at which the ρ-line reaches the level $\bar{f}$, i.e.

$$\mu = \frac{\bar{f} - f(0)}{\rho\, f'(0)} . \tag{3.65}$$

In the bracketing stage α_i is set in increasingly large jumps until a bracket $[a_i, b_i]$ on an interval of acceptable points is located. An algorithm suitable when objective function derivatives are available is given below.

Algorithm 3.3: Bracketing stage of line search.

Initially $\alpha_0 = 0$ and α_1 is given so that $0 < \alpha_1 \le \mu$. For each i the following iteration is repeated:

1. Evaluate $f(\alpha_i)$ and $f'(\alpha_i)$.
2. If $f(\alpha_i) \le \bar{f}$ then terminate line search.
3. If $f(\alpha_i) > f(0) + \alpha_i \rho f'(0)$ or $f(\alpha_i) \ge f(\alpha_{i-1})$ then
   set $\alpha_i = \alpha_{i-1}$ and $b_i = \alpha_i$, terminate bracketing.
4. If $|f'(\alpha_i)| \le -\sigma f'(0)$ then terminate line search.
5. If $f'(\alpha_i) \ge 0$ then
   set $a_i = \alpha_i$ and $b_i = \alpha_{i-1}$, terminate bracketing.
6. If $\mu \le \alpha_{i-1} + 2(\alpha_i - \alpha_{i-1})$ then set $\alpha_{i+1} = \mu$, else
   choose $\alpha_{i+1} \in [\alpha_{i-1} + 2(\alpha_i - \alpha_{i-1}), \min(\mu, \alpha_i + \tau_1(\alpha_i - \alpha_{i-1}))]$

τ_1 is a pre-set factor for which size of the jumps is increased, e.g. 10. Lines 2 to 5 terminate the bracketing stage, if a suitable bracket is located, or the whole line search, if an acceptable point or point for which $f(\alpha_i) \le \bar{f}$ is found. If neither of these situations take place in the current iteration, the search interval is extended (line 6). In this case it is convenient to choose α_{i+1} as a minimiser of some interpolation of $f(\alpha)$, e.g. a cubic polynomial constructed using $f(\alpha_{i-1})$, $f'(\alpha_{i-1})$, $f(\alpha_i)$ and $f'(\alpha_i)$.

If an acceptable point is not found in the bracketing stage, then a bracket $[a_i, b_i]$ is located, which contains an interval of acceptable points with respect to conditions (3.54) and (3.60). The bracket satisfies the following properties:

a. a_i is the current best trial point that satisfies (3.54) (it is possible that $b_i < a_i$, i.e. the bracket is not necessarily written with ordered extreme points).
b. $(b_i - a_i) f'(a_i) < 0$, but $f'(a_i)$ does not satisfy (3.60).
c. either $f(b_i) > f(0) + b_i \rho f'(0)$ or $f(b_i) \ge f(a_i)$ or both.

The sectioning stage is then performed in which the bracket is sectioned so that the length of subsequently generated brackets $\lfloor a_j, b_j \rfloor$ tend to zero. In each iteration a new trial point α_j is chosen and the next bracket is either $\lfloor a_j, \alpha_j \rfloor$, $\lfloor \alpha_j, a_j \rfloor$, or $\lfloor \alpha_j, b_j \rfloor$ or, so that the above described properties remain valid. The algorithm terminates when the current trial point α_j is acceptable with respect to (3.54) and (3.60).

Algorithm 3.4: Sectioning stage of the line search

A bracket $[a_0, b_0]$ is first available from the bracketing stage. The *j*-th iteration is then:

1. Choose $\alpha_j \in \lfloor a_j + \tau_2 (b_j - a_j), b_j - \tau_3 (b_j - a_j) \rfloor$,
   evaluate $f(\alpha_j)$ and $f'(\alpha_j)$.
2. If $f(\alpha_j) > f(0) + \rho \alpha_j f'(0)$ or $f(\alpha_j) \geq f(a_j)$, then
   set $a_{j+1} = a_j$ and $b_{j+1} = \alpha_j$, begin the next iteration.
3. If $\left| f'(\alpha_j) \leq -\sigma f'(0) \right|$, then terminate the line search.
4. Set $a_{j+1} = \alpha_j$
   If $(b_j - a_j) f'(\alpha_j) \geq 0$ then set $b_{j+1} = a_j$, else set $b_{j+1} = b_j$.

τ_1 and τ_2 are prescribed constants ($0 \leq \tau_1 \leq \tau_3 \leq \frac{1}{2},\ \tau_2 \leq \sigma$), which prevent α_j being arbitrarily close to a_j or b_j. Then

$$\left| b_{j+1} - a_{j+1} \right| \leq (1 - \tau_2) \left| b_j - a_j \right| \tag{3.66}$$

holds and the interval length therefore tends to zero. Their values can be for example $\tau_2 = \frac{1}{10}$ and $\tau_3 = \frac{1}{2}$. The choice of α_j in line 1 can again be made by minimisation of some interpolation of $f(\alpha)$.

If $\sigma > \rho$ then the algorithm terminates in a finite number of steps with α_j which is an acceptable point with respect to (3.54) and (3.60) [1].

In practice it can happen that the algorithm does not terminate because of numerical errors in the function and its derivatives. It is therefore advisable to terminate if $(a_j - \alpha_j) f'(a_j) \le \varepsilon$, where ε is some tolerance on *f*, with indication that no further progress can be made in the line search.

It is advantageous if a good initial choice α_1 (i.e. close to the line minimiser) can be made before the beginning of the bracketing stage. Some algorithms can give an estimation of likely reduction in the objective function in the line search Δf. This can be used in the quadratic interpolation of *f*, giving

$$\alpha_1 = -2 \frac{\Delta f}{f'(0)}. \tag{3.67}$$

A suitably safeguarded reduction in the previous iterate can be used as estimation of Δf, e.g. $\Delta f = \max\left(f^{(k-1)} - f^{(k)}, 10\varepsilon\right)$, where ε is the same tolerance as above. In the Newton-like methods (section 3.5) the choice $\alpha_1 = 1$ is significant in giving rapid convergence. Therefore the choice

$$\alpha_1 = \min\left(1, -2 \frac{\Delta f}{f'(0)}\right) \tag{3.68}$$

is usually made. The choice $\alpha_1 = 1$ is always made when iterates come close to the minimiser, if the method is superlinearly convergent.

3.4.3 Non-derivative Line Search Algorithms

If the line search is performed in an algorithm where derivatives are evaluated numerically by finite difference approximation, then $f'(\alpha)$ can also be approximated numerically and the line search strategy described in the previous section can be used. There also exist methods, which perform line searches, but do not use derivative information (e.g. direction set methods). In these methods non-derivative line search algorithms are used.

In the absence of derivative information, the criteria for acceptable points described in the previous section can not be applied. None-derivative line search methods rely on the fact that if we have three points *a*, *b* and *c* such that

$$a < b < c \wedge f(b) < f(a) \wedge f(b) < f(c), \tag{3.69}$$

then f has at least one local minimum in the interval $[a,c]$. It is then possible to section this interval, keeping three points, which satisfy the above relation through iterates.

The non-derivative line search also consists of a bracketing and sectioning stage. In the bracketing stage a triple of points $\{a_1, b_1, c_1\}$ that satisfy (3.69) is found in the following way:

Algorithm 3.5: Sectioning stage of a non-derivative line search.

Given $\alpha_0 = 0$, $f_0 = f(\alpha_0)$, α_1 and $f_1 = f(\alpha_1)$ such that $f_1 < f_0$, the *i*-th iteration is as follows:

1. Set $\alpha_i = \alpha_{i-1} + \zeta_1(\alpha_i - \alpha_{i-1})$, evaluate $f_i = f(\alpha_i)$.
2. If $f_i < \bar{f}$, accept α_i and terminate the line search.
3. If $f_i > f_{i-1}$, set $a_1 = \alpha_{i-2}$, $b_1 = \alpha_{i-1}$ and $c_1 = \alpha_i$, terminate bracketing.

Again $\bar{f}$ is some user defined value, so that the point with value of f lesser than $\bar{f}$ is automatically accepted as the line minimum. $\zeta_1 > 1$ is some factor which ensures that trial intervals are progressively enlarged, e.g. $\zeta_2 = 2$.

The algorithm assumes that initially $f_1 < f_0$. If $f_1 > f_0$, the algorithm can simply change α_0 and α_1 before the first iteration begins. The other possibility is to try with $\alpha' = -\alpha_1$. If $f' = f(\alpha') > f_0$, we can immediately terminate bracketing with $a_1 = \alpha'$, $b_1 = \alpha_0 = 0$ and $c_0 = \alpha_1$, otherwise we change α_0 and α_1 and set $\alpha_3 = \alpha'$.

The sectioning stage (Figure 3.7) is described below.

Algorithm 3.6: Sectioning stage of a non-derivative line search.

Given a triple of points $\{a_1, b_1, c_1\}$, which satisfy (3.69), the *j*-th iteration is as follows:

1. Choose $\alpha_j \in \left\{ \begin{matrix} \left[a_j + \zeta_2(b_j - a_j), b_j - \zeta_2(b_j - a_j)\right], \\ \left[b_j + \zeta_2(c_j - b_j), c_j - \zeta_2(c_j - b_j)\right] \end{matrix} \right\}$, evaluate $f^{(j)} = (\alpha_j)$.
2. If $f_j < \bar{f}$, then accept α_j and terminate the line search.
3. If $\alpha_j \in [b_{j-1}, c_{j-1}]$, interchange a_j and c_j.

4. If $\alpha_j < b_j$, set $a_{j+1} = a_j$, $b_{j+1} = \alpha_j$ and $c_{j+1} = b_j$, go to line 6.
5. If $\alpha_j \geq b_j$, set $a_{j+1} = \alpha_j$, $b_{j+1} = b_j$ and $c_{j+1} = c_j$, go to line 6.
6. Check convergence criterion. If the criterion is satisfied, then terminate with b_j as the line minimum.

A triple of points $\{a_j, b_j, c_j\}$ satisfies the condition (3.69) through all iterations. Parameter ζ_2 ($0 < \zeta_2 < \frac{1}{2}$) ensures that the lengths of successive brackets $[a_j, c_j]$ tend to zero. $\zeta_2 = 0.1$ is a reasonable choice.

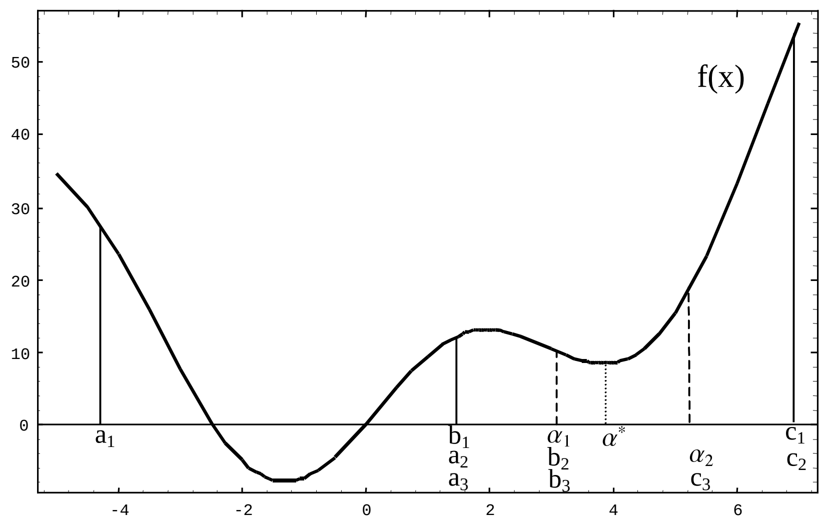

Figure 3.7: Sectioning stage of the non-derivative line search in the case when interpolation is not applied.

α_j can be chosen as the minimiser of a quadratic interpolation of *f*, i.e. a parabola through the points $(a_j, f(a_j))$, $(b_j, f(b_j))$ and $(c_j, f(c_j))$. The formula of such parabola is

$$p(\alpha) = f(a)\frac{(x-b)(x-c)}{(a-b)(a-c)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)}$$

and its minimiser is

$$\alpha_{\min} = b + \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)]-(b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)]-(b-c)[f(b)-f(a)]}, \tag{3.70}$$

where indices *j* have been omitted.

If $a_{\min}$ is an element of the acceptable interval in line 1 of the above algorithm, then $\alpha_j = \alpha_{\min}$ is set. Otherwise the longer α_j is obtained by sectioning the longer of both intervals. If the longer interval is $[a_j, b_j]$, then

$$\alpha_j = a_j + \frac{\tau}{1+\tau}(b_j - a_j), \tag{3.71}$$

where τ is some fixed parameter such that $0.5 < \tau < 1-\zeta_2$.

A common choice is the golden section ratio $\tau = (1+\sqrt{5})/2 \approx 1.618$. It follows from the request that when a new point α_j is inserted in $[a_j, b_j]$, both potentially new brackets have the same interval length ratio $\frac{c_{j+1}-b_{j+1}}{b_{j+1}-a_{j+1}}$ (i.e. $1/\tau$), which then gives $\tau = (1+\sqrt{5})/2$. This request can be applied when pure bracketing takes place and also the initial triple has the same interval length ratio.

The convergence can be checked either on function values, e.g.

$$\max\{(f(a_j)-f(b_j)),(f(c_j)-f(b_j))\} < \varepsilon \tag{3.72}$$

or on interval length, i.e.

$$|c_j - a_j| < \varepsilon. \tag{3.73}$$

3.5 Newton-like Methods

Newton-like methods are based on a quadratic model, more exactly on the second-order Taylor approximation (equation) of $f(\mathbf{x})$ about $\mathbf{x}^{(k)}$. The basic ideas around this were explained in sections 3.1.2 and 3.3 and will be further developed in this section.

In section 3.1.2 Newton's method was derived from the solution of the system of equations

$$\nabla \mathbf{g}(\mathbf{x}) = 0\,,$$

where the iteration formula was derived from the first order Taylor's approximation of $\mathbf{g}(\mathbf{x})$, giving iteration formula (3.5). Two problems related with direct application of the method were mentioned there, i.e. lack of global convergence properties and explicit use of the second order derivative information regarding the objective functions. Some general ideas on how to overcome these problems were outlined in section 3.3 and will be further developed in this section for algorithms, which in principle stick with the basic idea of Newton's method.

In order to take over and develop the ideas given in section 3.3, let us start from the second order Taylor approximation of f itself, developed around the current iterate:

$$f\left(\mathbf{x}^{(k)} + \delta\right) \approx q^{(k)}(\delta) = f^{(k)} + \mathbf{g}^{(k)T}\delta + \frac{1}{2}\delta^T \mathbf{G}^{(k)}\delta\,. \tag{3.74}$$

Using the results of section 3.3, the stationary point of this approximation is a solution of a linear system of equations

$$\mathbf{G}^{(k)}\delta = -\mathbf{g}^{(k)}\,. \tag{3.75}$$

It is unique if $\mathbf{G}^{(k)}$ is non-singular and corresponds to a minimiser if $\mathbf{G}^{(k)}$ is positive definite. Newton's method is obtained by considering $\delta^{(k)}$ as solution of the above equation and setting the next guess to $\mathbf{x}^{(k)} + \delta^{(k)}$. The k-th iteration of Newton's method is then

1. Solve (3.75) for $\delta^{(k)}$,
2. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$.

This is well defined as a minimisation method only if $\mathbf{G}^{(k)}$ is positive definite in each iteration, and this can be readily checked if for example LDLT decomposition is used for solution of (3.75). However, even if $\mathbf{G}^{(k)}$ is positive definite, the method may not converge from any initial guess, and it can happen that $\left\{f^{(k)}\right\}$ do not even decrease.

Line search can be used to eliminate this problem. The solution of (3.75) then defines merely the search direction, rather than correction $\delta^{(k)}$. The correction is then obtained by line minimisation using algorithms described in section 3.4.2, and such a method is called Newton's method with line search. The direction of search is

$$\mathbf{s}^{(k)} = -\mathbf{G}^{(k)-1}\mathbf{g}^{(k)}. \qquad (3.76)$$

If $\mathbf{G}^{(k)}$ and hence its inverse are positive definite, this defines a descent direction. If $\mathbf{G}^{(k)}$ is not positive definite, it may be possible to make a line search in $\pm\mathbf{s}^{(k)}$, but the relevance of searching in $-\mathbf{s}^{(k)}$ is questionable because this is not a direction towards a stationary point of $q(\delta)$. Furthermore, the method fails if any $\mathbf{x}^{(k)}$ is a saddle point of *f*. This gives $\mathbf{s}^{(k)} = 0$, although $\mathbf{x}^{(k)}$ is not a minimiser of *f*.

One possibility of how to overcome this problem is to switch to the steepest descent direction whenever $\mathbf{G}^{(k)}$ is not positive definite. This can be done in conjunction with the angle criterion (3.61) to achieve global convergence.

Minimising in the steepest descent directions can lead to undesired oscillatory behaviour where small reductions of *f* are achieved in each iteration. This happens because second order model information is ignored, as shown in section 3.3.3. The alternative approach is to switch between the Newton and steepest descent direction in a continuous way, controlling the influence of both through some adaptive weighting parameter. This can be achieved by adding a multiple of the unit matrix to $\mathbf{G}^{(k)}$ so that the search direction is defined as

$$\left(\mathbf{G}^{(k)} + \nu\,\mathrm{I}\right)\mathbf{s}^{(k)} = -\mathbf{g}^{(k)}. \qquad (3.77)$$

Parameter ν is chosen so that $\mathbf{G}^{(k)} + \nu\,\mathrm{I}$ is positive definite. If $\mathbf{G}^{(k)}$ is close to positive definite, a small ν is sufficient and the method therefore uses the curvature information to a large extent. When large values of ν are necessary, the search directions becomes similar to the steepest descent direction $-\mathbf{g}^{(k)}$.

This method still fails when some $\mathbf{x}^{(k)}$ is a saddle point, and the second order information is not used in the best possible way. Further modification of the method incorporates the restricted step approach in which minimisation of the model quadratic function subjected to length restriction is minimised. This is a subject of section 3.7.

3.5.1 Quasi-Newton Methods

In the Newton-like methods discussed so far the second derivatives of f are necessary and substantial problems arise when the Hessian matrix of the function is not positive definite. The second derivatives of $\mathbf{G}^{(k)}$ can be evaluated by numerical differentiation of the gradient vector. In most cases it is advisable that after this operation $\mathbf{G}$ is made symmetric by $\mathbf{G} = \frac{1}{2}\left(\overline{\mathbf{G}} + \overline{\mathbf{G}}^T\right)$, where $\overline{\mathbf{G}}$ is the finite difference approximation of the Hessian matrix. However, evaluation of $\mathbf{G}$ can be unstable in the presence of numerical noise, and it is also expensive, because quadratic model information built in the previous iterates is disregarded.

The above mentioned problems are avoided in so called quasi-Newton methods. In these methods $\mathbf{G}^{(k)-1}$ are approximated by symmetric matrices $\mathbf{H}^{(k)}$, which are updated from iteration to iteration using the most recently obtained information. Analogous to Newton's method with line search, line minimisations are performed in each iteration in the direction

$$\mathbf{s}^{(k)} = -\mathbf{H}^{(k)}\mathbf{g}^{(k)}. \tag{3.78}$$

By updating approximate $\mathbf{G}^{-1}$ rather than $\mathbf{G}$, a system of equations is avoided and the search direction is obtained simply by multiplication of the gradient vector by a matrix. An outline of the algorithm is given below:

Algorithm 3.7: General quasi-Newton algorithm.

Given a positive definite matrix $\mathbf{H}^{(1)}$, the k-th iteration is:

1. Calculate $\mathbf{s}^{(k)}$ according to (3.78).
2. Minimise f along $\mathbf{s}^{(k)}$, set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{s}^{(k)}$, where $\alpha^{(k)}$ is a line minimum.
3. Update $\mathbf{H}^{(k)}$ to obtain $\mathbf{H}^{(k+1)}$.

If no second derivative information is available at the beginning, $\mathbf{H}^{(1)}$ can be any positive definite matrix, e.g. $\mathbf{H}^{(1)} = \mathbf{I}$. The line search strategy described in section 3.4.2 can be used in line 2. If $\mathbf{H}^{(k)}$ is positive definite, the search directions are descent. This is desirable and the most important are those quasi-Newton methods, which maintain positive definiteness of $\mathbf{H}^{(k)}$.

The updating formula should explicitly use only first derivative information. Repeated updating should change arbitrary $\mathbf{H}^{(1)}$ to a close approximation of $\mathbf{G}^{(k)-1}$ The updating formula is therefore an attempt to augment the current $\mathbf{H}^{(k)}$ with second derivative information gained in the current iteration, i.e. by evaluation of f and ∇f at two distinct points. In this context equation (3.27), which relates the Hessian matrix of a quadratic function with its gradient in two distinct points, requires attention.

Let us write

$$\delta^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \tag{3.79}$$

and

$$\gamma^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}. \tag{3.80}$$

Using the Taylor series of **g** about $\mathbf{x}^{(k)}$ gives a relationship similar to (3.27), i.e.

$$\gamma^{(k)} = \mathbf{G}^{(k)}\delta^{(k)} + o\left(\left\|\delta^{(k)}\right\|\right). \tag{3.81}$$

The updating formula should therefore correct $\mathbf{H}^{(k+1)}$ so that the above relation would hold approximately with $\mathbf{H}^{(k+1)-1}$ in place of $\mathbf{G}^{(k)}$. This gives the so called quasi-Newton condition, in which the updating formula must satisfy

$$\mathbf{H}^{(k+1)}\gamma^{(k)} = \delta^{(k)}. \tag{3.82}$$

Since this condition gives only one equation, it does not uniquely define the updating formula and permits various ways of updating **H**. One possibility is to add a symmetric rank one matrix to $\mathbf{H}^{(k)}$, i.e.

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{u}\mathbf{u}^T. \tag{3.83}$$

Substituting this into (3.82) gives

$$\mathbf{H}^{(k)}\gamma^{(k)} + \mathbf{u}\mathbf{u}^T\gamma^{(k)} = \delta^{(k)}. \tag{3.84}$$

Since $\mathbf{u}^{(T)}\gamma^{(k)}$ is a scalar, matrix multiplication is associative and multiplication with a scalar is commutative, **u** must be proportional to $\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}$. Writing

$$\mathbf{u} = a\left(\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}\right)$$

and inserting this into (3.84) gives $a = 1\Big/\sqrt{\left(\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}\right)^{T}\gamma^{(k)}}$ and therefore

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \frac{\left(\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}\right)\left(\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}\right)^{(T)}}{\left(\delta^{(k)} - \mathbf{H}^{(k)}\gamma^{(k)}\right)^{(T)}\gamma^{(k)}}. \tag{3.85}$$

This formula is called the rank one updating formula according to the above derivation.

For a quadratic function with positive definite Hessian the rank one method terminates in at most n+1 steps with $\mathbf{H}^{(n+1)} = \mathbf{G}^{-1}$, provided that $\delta^{(1)}, ..., \delta^{(n)}$ are independent and that the method is well defined[1]. The proof does not require exact line searches. Also the so called hereditary property can be established, i.e.

$$\mathbf{H}^{(i)}\gamma^{(j)} = \delta^{(j)}, \quad j = 1, 2, ..., i-1. \tag{3.86}$$

A disadvantage is that in general the formula does not maintain positive definiteness of $\mathbf{H}^{(k)}$ and the dominator in (3.85) can become zero.

Better formulas can be obtained by allowing the correction to be of rank two. This can always be written[29],[31] as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{u}\mathbf{u}^{T} + \mathbf{v}\mathbf{v}^{T}. \tag{3.87}$$

Using this in the quasi-Newton condition gives

$$\delta^{(k)} = \mathbf{H}^{(k)}\gamma^{(k)} + \mathbf{u}\mathbf{u}^{T}\gamma^{(k)} + \mathbf{v}\mathbf{v}^{T}\gamma^{(k)}. \tag{3.88}$$

u and **v** can not be determined uniquely. A straightforward way of satisfying the above equation is to set **u** proportional to $\delta^{(k)}$ and **v** proportional to $\mathbf{H}^{(k)}\gamma^{(k)}$. By solution of the equation separately for both groups of proportional vectors the Davidon – Fletcher - Powell or DFP updating formula is obtained:

$$\mathbf{H}_{DFP}^{(k+1)} = \mathbf{H} + \frac{\delta\delta^{T}}{\delta^{T}\gamma} - \frac{\mathbf{H}\gamma\gamma^{T}\mathbf{H}}{\gamma^{T}\mathbf{H}\gamma}. \tag{3.89}$$

Indices k have been omitted for the sake of simplicity (this approach will be adopted through this section) and the symmetry of **H** is used.

Another rank two updating formula can be obtained by considering updating and approximating $\mathbf{G}$ instead of $\mathbf{G}^{-1}$. Let us write $\mathbf{B}^{(k)} = \mathbf{H}^{(k)-1}$ and consider updating $\mathbf{B}^{(k)}$ in a similar way as $\mathbf{H}^{(k)}$ was updated according to the DFP formula. We require that the quasi-Newton condition (3.82) is preserved. This was true for the DFP formula, but now we are updating inverse of $\mathbf{H}^{(k)}$, therefore, according to (3.82), $\gamma^{(k)}$ and $\delta^{(k)}$ must be interchanged. This gives the formula

$$\mathbf{B}_{BFGS}^{(k+1)} = \mathbf{B} + \frac{\gamma\gamma^T}{\gamma^T\delta} - \frac{\mathbf{B}\delta\delta^T\mathbf{B}}{\delta^T\mathbf{B}\delta}. \tag{3.90}$$

We however still want to update $\mathbf{H}^{(k)}$ rather than $\mathbf{B}^{(k)}$, because a solution of system of equations is in this way avoided in the quasi-Newton iteration. The following updating formula satisfies $\mathbf{B}_{BFGS}^{(k+1)}\mathbf{H}_{BFGS}^{(k+1)} = \mathbf{I}$:

$$\mathbf{H}_{BFGS}^{(k+1)} = \mathbf{H} + \left(1 + \frac{\gamma^T\mathbf{H}\gamma}{\delta^T\gamma}\right)\frac{\delta\delta^T}{\delta^T\gamma} - \left(\frac{\delta\gamma^T\mathbf{H} + \mathbf{H}\gamma\delta^T}{\delta^T\gamma}\right). \tag{3.91}$$

This is called the Broyden – Fletcher – Goldfarb – Shanno or BFGS updating formula.

The BFGS and the DFP formula are said to be dual or complementary because the expressions for $\mathbf{B}^{(k+1)}$ and $\mathbf{H}^{(k+1)}$ in one are obtained by interchanging $\mathbf{B} \leftrightarrow \mathbf{H}$ and $\gamma \leftrightarrow \delta$ in the other. Such duality transformation preserves the quasi-Newton condition. The rank one formula is self-dual.

The DFP and BFGS updating formula can be combined to obtain the so called Broyden one-parameter family of rank two formulae:

$$\mathbf{H}_{\phi}^{k+1} = (1-\phi)\mathbf{H}_{DFP}^{(k+1)} + \phi\,\mathbf{H}_{BFGS}^{(k+1)}. \tag{3.92}$$

This family includes the DFP and BFGS and also rank 1 formula. The quasi-Newton method with a Broyden family updating formula has the following properties[1]:

1. For a quadratic function with exact line searches:
- The method terminates in at most *n* iterations with $\mathbf{H}^{(n+1)} = \mathbf{G}^{-1}$.
- Previous quasi-Newton conditions are preserved (hereditary property (3. 86)).
- Conjugate directions are generated, and conjugate gradients when $\mathbf{H}^{(0)} = \mathbf{I}$.
2. For general functions:
- The method has superlinear order of convergence.
- The method is globally convergent for strictly convex functions if exact line searches are performed.

The Broyden family updates maintain positive definiteness of $\mathbf{H}_{\phi}^{(k+1)}$ for $\phi \geq 0$.

Global convergence has also been proved for the BFGS method with inexact line searches subject to conditions (3.56) and (3.59), applied to a convex objective function[1]. The BFGS method with inexact line searches converges superlinearly if $\mathbf{G}^{(*)}$ is positive definite.

The BFGS method also shows good performance in numerical experiments. The method is not sensitive to exactness of line searches, in fact it is a generally accepted opinion that inexact line searches are more efficient with the BFGS method than near exact line searches. The contemporary optimisation literature[1],[4] suggests the BFGS method as preferable choice for general unconstrained optimisation based on a line search prototype algorithm.

3.5.2 Invariance Properties

It is important to study how optimisation algorithms perform when affine transformation of variables is made, i.e.

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{a}, \tag{3.93}$$

where **A** is nonsingular. This is a one-to-one mapping with inverse transformation

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{a}).$$

f can be evaluated either in **x** space (denoted by $f_x(\mathbf{x})$) or in **y** space (denoted by $f_y(\mathbf{y}) = f_x\left(\mathbf{A}^{-1}(\mathbf{y} - \mathbf{a})\right)$).

Applying the chain rule for derivation in **x** space gives

$$\frac{\partial}{\partial x_i} = \sum_{k=1}^{n} \frac{\partial y_k}{\partial x_i} \frac{\partial}{\partial y_k} = \sum_{k=1}^{n} \left(\mathbf{A}^T\right)_{ik} \frac{\partial}{\partial y_k}, \tag{3.94}$$

therefore $\nabla_x = \mathbf{A}^T \nabla_y$ and so

$$\mathbf{g}_x = \mathbf{A}^T \mathbf{g}_y. \tag{3.95}$$

Applying the gradient operator to the above equation then gives $\nabla_x \mathbf{g}_x{}^T = \mathbf{A}^T \nabla_y \mathbf{g}_y{}^T \mathbf{A}$, i.e.

$$\mathbf{G}_x = \mathbf{A}^T \mathbf{G}_y \mathbf{A} . \tag{3.96}$$

The notation $\mathbf{g}_y = \nabla_y f_y$, etc. was used, so that for example

$$[\mathbf{G}_y]_{ij} = \frac{\partial^2 f_y}{\partial y_i \partial y_j} .$$

The following theorem[1] applies to Newton-like methods:

Theorem 3.4:

If $\mathbf{H}^{(k)}$ transforms under transformation (3.93) as

$$\mathbf{H}_x^{(k)} = \mathbf{A}^{-1} \mathbf{H}_y^{(k)} \mathbf{A}^{-T} \quad \forall k , \tag{3.97}$$

then a Newton-like method with fixed step $\alpha^{(k)}$ is invariant under the transformation (3.93). A method is also invariant if $\alpha^{(k)}$ is determined by tests on $f^{(k)}$, $\mathbf{g}^{(k)T} \mathbf{s}^{(k)}$ or other invariant scalars.

Transformation (3.97) in the above theorem is obtained by inverting (3.96), since $\mathbf{H}^{(k)}$ approximate $\mathbf{G}^{(k)}$ in the quasi-Newton methods.

We see that the steepest descent method (treated as quasi-Newton method with $\mathbf{H}^{(k)} = \mathbf{I}$) is not invariant under transformation (3.93) because $\mathbf{I}$ does not transform correctly. Modified Newton methods are also not invariant because $\mathbf{G} + \nu \mathbf{I}$ does not transform correctly when $\nu > 0$.

For a quasi-Newton method to be invariant, $\mathbf{H}^{(1)}$ must be chosen so as to transform correctly (as (3.97)) and the updating formula must preserve the transformation property (3.97). Therefore, if $\mathbf{H}^{(1)} = \mathbf{I}$ is chosen, then invariance does not hold. $\mathbf{H}^{(1)} = \mathbf{G}\left(\mathbf{x}^{(1)}\right)^{-1}$ transforms correctly and therefore this choice does not affect invariance.

In order to show that a specific updating formula preserves the transformation property (3.97), we must show that $\mathbf{A}\mathbf{H}_x^{(k)}\mathbf{A}^T = \mathbf{H}_y^{(k)}$ (which is (3.97) pre-multiplied by $\mathbf{A}$ and post-multiplied by $\mathbf{A}^T$) which implies $\mathbf{A}\mathbf{H}_x^{(k+1)}\mathbf{A}^T = \mathbf{H}_y^{(k+1)}$. Let us do this for the DFP formula

$$\mathbf{H}_x^{(k+1)} = \mathbf{H}_x + \frac{\delta_x \delta_x{}^T}{\delta_x{}^T \gamma_x} - \frac{\mathbf{H}_x \gamma_x \gamma_x{}^T \mathbf{H}_x}{\gamma_x{}^T \mathbf{H}_x \gamma_x}. \tag{3.98}$$

We will pre-multiply the above equation by $\mathbf{A}$ and post-multiply it by $\mathbf{A}^T$ and use relations $\mathbf{A}\delta_x = \delta_y$ following from (3.93) and $\gamma_x = \mathbf{A}^T \gamma_y$ following from (3.95). We will consider individual terms in equation (3.98).

The first term on the right-hand side of (3.98) gives, after multiplication, $\mathbf{H}_y^{(k)}$ by assumption. Consider then the denominator of the second term:

$$\delta_x{}^T \gamma_x = \delta_x{}^T \mathbf{A}^T \mathbf{A}^{-T} \gamma_x = \left(\mathbf{A}\delta_x\right)^T \gamma_y = \delta_y{}^T \gamma_y,$$

the denominator is invariant. The numerator after multiplication gives

$$\mathbf{A}\delta_x \delta_x{}^T \mathbf{A}^T = \delta_y \delta_y{}^T,$$

so the second term transforms correctly. Consider the denominator of the third term:

$$\gamma_x{}^T \mathbf{H}_x \gamma_x = \gamma_x{}^T \mathbf{A}^{-1} \mathbf{A} \mathbf{H}_x \mathbf{A}^T \mathbf{A}^{-T} \gamma_x = \left(\mathbf{A}^{-T}\gamma_x\right)^T \mathbf{H}_y \mathbf{A}^{-T} \gamma_x = \gamma_y{}^T \mathbf{H}_y \gamma_y,$$

the denominator is invariant under transformation. The numerator after multiplication is

$$\mathbf{A}\mathbf{H}_x \gamma_x \gamma_x{}^T \mathbf{H}_x \mathbf{A}^T = \mathbf{A}\mathbf{H}_x \mathbf{A}^T \mathbf{A}^{-T} \gamma_x \gamma_x{}^T \mathbf{A}^{-1} \mathbf{A}\mathbf{H}_x \mathbf{A}^T = \mathbf{H}_y \gamma_y \left(\mathbf{A}^{-T}\gamma_x\right)^T \mathbf{H}_y = \mathbf{H}_y \gamma_y \gamma_y{}^T \mathbf{H}_y,$$

so the third term is also transformed correctly. $\mathbf{A}\mathbf{H}_x^{(k+1)}\mathbf{A}^T = \mathbf{H}_y^{(k+1)}$ is valid since

$$\mathbf{A}\mathbf{H}_x^{(k+1)}\mathbf{A}^T = \mathbf{H}_y + \frac{\delta_y \delta_y{}^T}{\delta_y{}^T \gamma_y} - \frac{\mathbf{H}_y \gamma_y \gamma_y{}^T \mathbf{H}_y}{\gamma_y{}^T \mathbf{H}_y \gamma_y}$$

and this is the DFP formula in the **y** space.

Similarly the preservation of (3.97) can be proved for all updating formulas in which the correction is a sum of rank one terms constructed from vectors δ and $\mathbf{H}\gamma$, multiplied by invariant scalars. Such versions are the BFGS formula and hence all Broyden family formulas.

The Broyden family (including BFGS and DFP) algorithms are therefore invariant under the affine transformation of variables (3.93), provided that $\mathbf{H}^{(1)}$ is chosen so as to transform correctly, i.e. as (3.97). However, even if $\mathbf{H}^{(1)}$ is not chosen correctly, after *n* iterations we have $\mathbf{H}^{(n+1)} \approx \mathbf{G}^{(n+1)-1}$, which is transforms correctly. The method therefore becomes close to the one in which invariance is preserved.

Invariance to an affine transformation of variables is a very important algorithmic property. Algorithms which have this property, are less sensitive to situations in which **G** is ill-conditioned, since an implicit transformation which transforms **G** to the unity matrix **I** can be introduced, which does not change the method. Algorithms that are not invariant, i.e. the steepest descent or the alternating variables method, can perform very badly when the Hessian is ill-conditioned.

When using methods which are not invariant, it can be advantageous to find a linear transformation which improves the conditioning of the problem[14].

If columns of **A** are eigenvectors of **G**, then **G** is diagonalised when transformation (3.96) is applied. Conditioning can be achieved by additional scaling of variables, i.e. by multiplication with a diagonal matrix. This approach is however not applicable in practice because it is usually difficult to calculate eigenvectors of **G**. For positive definite **G** the same effect is achieved by using Choleski factors of **G** as the transformation matrix. $\mathbf{G}_x = \mathbf{A}^T\mathbf{A}$ gives

$$\mathbf{G}_y = \mathbf{A}^{-T}\mathbf{G}_x\mathbf{A}^{-1} = \mathbf{A}^{-T}\mathbf{A}^T\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}\,.$$

It is often possible to improve conditioning just by scaling the variables. In this case **A** is chosen to be a diagonal matrix so that $\mathbf{A}^2$ estimates $\mathbf{G}_x$ in some sense. $\mathbf{G}_y = \mathbf{A}^{-T}\mathbf{G}_x\mathbf{A}^{-1}$ (from (3.96)) is required to be close to the unity matrix in some sense. It can be required, for example, that $\left[\mathbf{G}_y\right]_{ii} = 1\,\forall i$. It is usually not necessary to explicitly perform the scaling, but **I** can be replaced in the methods by a suitable diagonal matrix. For example, the modified Newton method can be improved by using $\mathbf{G} + \nu\mathbf{A}^{-2}$ in place of $\mathbf{G} + \nu\,\mathrm{I}$.

3.6 Conjugate Direction Methods

Optimisation algorithms described in this section are based on the result given in Theorem 3.2, which associates conjugacy and exact line searches with quadratic termination. These algorithms rely on an idealized assumption that exact line searches are performed as in Algorithm 3.2. This is possible for a quadratic function, but not in general. By using interpolation in the line search algorithm, it is still possible to locate a local minimum up to a certain accuracy, and this approach is used in practice with the conjugate direction methods. An argument which justifies this is that in the close neighbourhood of a minimum, quadratic interpolations of the objective functions will enable the line minimum to be located almost exactly, so that the inexact nature of the line search algorithm will not spoil local convergence properties, which are theoretically based on the assumption of exact line search.

In section 3.6.1 derivative based conjugate direction methods are described. In section 3.6.2 algorithms based on the idea of conjugacy, but in the absence of derivative information are treated. All the described methods generate conjugate directions when they are applied to a quadratic function.

3.6.1 Conjugate Gradient Methods

Conjugate gradient methods begin with line search along

$$\mathbf{s}^{(1)} = -\mathbf{g}^{(1)} \tag{3.99}$$

and then generate search directions $\mathbf{s}^{(k+1)}, k \geq 1$ from $-\mathbf{g}^{(k+1)}$, so that they are conjugate to $\mathbf{s}^{(1)}, ..., \mathbf{s}^{(k)}$ with respect to the Hessian matrix $\mathbf{G}$ when a method is applied to a quadratic function.

For a quadratic function it follows from (3.27) that

$$\gamma^{(k)} = \mathbf{G}\delta^{(k)} \, \forall k , \tag{3.100}$$

where $\gamma^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$ and $\delta^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, as usual. Conjugacy conditions (3.43) can therefore be written as

$$\mathbf{s}^{(i)T}\gamma^{(j)} = 0 \quad j \neq i \tag{3.101}$$

since $\gamma^{j} = \mathbf{G}\delta^{(j)} = \mathbf{G}\alpha^{(j)}\mathbf{s}^{(j)}$. The last expression is a consequence of the fact that $\mathbf{x}^{(j+1)}$ is obtained by a line search performed from $\mathbf{x}^{(j)}$ along $\mathbf{s}^{(j)}$.

The above equation can be used to prove an important property. First we can see that

$$\mathbf{s}^{i^T}\mathbf{g}^{(i+1)} = 0 \;\; \forall i\,, \tag{3.102}$$

because exact line searches are used. By using the above equation and (3.101) we obtain

$$\begin{aligned}
&\mathbf{s}^{(i)T}\mathbf{g}^{(k+1)} = \\
&\mathbf{s}^{(i)(T)}\left(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)} + \mathbf{g}^{(k)} - \mathbf{g}^{(k-1)} + \ldots - \mathbf{g}^{(i+1)} + \mathbf{g}^{(i+1)}\right) = , \\
&\mathbf{s}^{(i)(T)}\left(\gamma^{(k)} + \gamma^{(k)} + \ldots \gamma^{(i+1)} + \mathbf{g}^{(i+1)}\right) = 0 \;\; \forall i, k > i
\end{aligned} \tag{3.103}$$

This means that $\mathbf{g}^{(k+1)}$ is orthogonal to all search directions of previous steps:

$$\mathbf{s}^{(i)T}\mathbf{g}^{(k+1)} = 0 \;\; \forall k, i \leq k\,. \tag{3.104}$$

This is actually the result of Theorem 3.2.

In the Fletcher-Reeves method $\mathbf{s}^{(k+1)}$ is obtained from $-\mathbf{g}^{(k+1)}$ by the extended Gramm-Schmidt orthogonalisation[27],[29] with respect to $\gamma^{(i)},\, j \leq k$, in order to satisfy conjugacy conditions (3.101). We can write[1]

$$\mathbf{s}^{(k+1)} = -\mathbf{g}^{(k+1)} + \sum_{j=1}^{k} \beta^{(j)}\mathbf{s}^{(j)}\,. \tag{3.105}$$

Multiplying the transpose of the above equation by $\gamma^{(i)}$ gives

$$\mathbf{s}^{(k+1)T}\gamma^{(i)} = 0 = -\mathbf{g}^{(k+1)T}\gamma^{(i)} + \beta^{(i)}\mathbf{s}^{(i)T}\gamma^{(i)}\,, \tag{3.106}$$

[1] The derivation of the Fletcher-Reeves method was found to be not completely clear in some optimisation literature and is therefore included herein.

where (3.101) was taken into account. It follows that

$$\beta^{(i)} = \frac{\mathbf{g}^{(k+1)T}\gamma^{(i)}}{\mathbf{s}^{(i)T}\gamma^{(i)}} = \frac{\mathbf{g}^{(k+1)T}\left(\mathbf{g}^{(i+1)} - \mathbf{g}^{(i)}\right)}{\mathbf{s}^{(i)T}\left(\mathbf{g}^{(i+1)} - \mathbf{g}^{(i)}\right)}. \tag{3.107}$$

It follows from construction of $\mathbf{s}^{(k)}$ ((3.99) and (3.105)) that vectors $\mathbf{g}^{(1)}, ..., \mathbf{g}^{(k)}$ and $\mathbf{s}^{(1)}, ..., \mathbf{s}^{(k)}$ span the same subspace. Therefore, since $\mathbf{g}^{(k+1)}$ is orthogonal to the subspace spanned by $\mathbf{s}^{(1)}, ..., \mathbf{s}^{(k)}$ due to (3.104), it is also orthogonal to vectors $\mathbf{g}^{(1)}, ..., \mathbf{g}^{(k)}$, i.e.

$$\mathbf{g}^{(i)T}\mathbf{g}^{(k+1)} = 0 \quad \forall k, i \le k \tag{3.108}$$

We see that only $\beta^{(k)} \neq 0$ and that

$$\beta^{(k)} = \frac{\mathbf{g}^{(k+1)T}\left(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right)}{\mathbf{s}^{(k)T}\left(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right)} = -\frac{\mathbf{g}^{(k+1)T}\mathbf{g}^{(k+1)}}{\mathbf{s}^{(k)T}\mathbf{g}^{(k)}} \tag{3.109}$$

The denominator of the above equation can be obtained by substituting $\mathbf{s}^{(k)}$ by (3.105) with decreased indices and taking into account that only $\beta^{(k-1)}$ is non-zero, together with the established orthogonality properties:

$$\mathbf{s}^{(k)T}\mathbf{g}^{(k)} = \left(-\mathbf{g}^{(k)} + \beta^{(k-1)}\mathbf{s}^{(k-1)}\right)^T \mathbf{g}^{(k)} = -\mathbf{g}^{(k)T}\mathbf{g}^{(k)}.$$

Now we have

$$\beta^{(k)} = \frac{\mathbf{g}^{(k+1)T}\mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}. \tag{3.110}$$

The obtained results can be summarized in the following way:

Theorem 3.5:

The Fletcher-Reeves method with exact line searches terminates for a quadratic function at a stationary point $\mathbf{x}^{m+1}$ after $m \le n$ iterations. In addition, the following results hold for $1 \le i \le m$:

$$\mathbf{s}^{(i)T}\mathbf{G}\mathbf{s}^{(j)} = 0; \quad j = 1, 2, ..., i-1 \text{ (conjugate directions)}, \tag{3.111}$$

$$\mathbf{g}^{(i)T}\mathbf{g}^{(j)} = 0;\quad j = 1, 2, ..., i-1 \text{ (orthogonal gradients)} \tag{3.112}$$

and

$$\mathbf{s}^{(i)T}\mathbf{g}^{(i)} = -\mathbf{g}^{(i)T}\mathbf{g}^{(i)} \text{ (descent conditions).} \tag{3.113}$$

The termination must occur in at least n iterations because in the opposite case $\mathbf{g}^{(n+1)} \neq 0$ would contradict the result that gradients are orthogonal.

When applied to a quadratic function with positive definite **G**, the Fletcher-Reeves method turns to be equivalent to the Broyden family of methods if $\mathbf{H}^{(1)} = \mathbf{I}$, the starting point is the same and exact line searches are performed in both methods[1],[4]. For non-quadratic functions line search Algorithm 3.4 is recommended with $\sigma = 0.1$. Resetting the search direction to the steepest descent direction periodically after every n iterations is generally an accepted strategy in practice. When compared with quasi-Newton methods, conjugate gradient methods are less efficient and less robust and they are more sensitive to the accuracy of the line search algorithm. Methods with resetting are globally convergent and exhibit n-step superlinear convergence, i.e.

$$\lim_{k\to\infty} \frac{\left\| \mathbf{x}^{(k+n)} - \mathbf{x}^* \right\|}{\left\| \mathbf{x}^{(k)} - \mathbf{x}^* \right\|} = 0 \tag{3.114}$$

Some other formulas may be used instead of (3.110). Examples are the conjugate descent formula

$$\beta^{(k)} = \frac{\mathbf{g}^{(k+1)T}\mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)T}\mathbf{s}^{(k)}} \tag{3.115}$$

and the Polak-Ribiere formula

$$\beta^{(k)} = \frac{\left(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}\right)^T \mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)T}\mathbf{g}^{(k)}}. \tag{3.116}$$

Considering the derivation of the Fletcher-Reeves method, it can be seen that these formulas are equivalent to the Fletcher-Reeves formula when applied to quadratic functions with exact line searches. The conjugate descent formula has a strong descent property that $\mathbf{s}^{(k)T}\mathbf{g}^{(k)} < 0$ if $\mathbf{g}^{(k)} \neq 0$. The Polak-Ribiere formula is recommended when solving large problems[1].

Another possibility for conjugate gradient methods is to use symmetric projection matrices in the calculation of $\mathbf{s}^{(k+1)}$, which annihilate vectors $\gamma^{(k)}, ..., \gamma^{(k)}$:

$$\mathbf{s}^{(k)} = -\mathbf{P}^{(k)}\mathbf{g}^{(k)}, k = 1, 2, ..., n. \tag{3.117}$$

Initially

$$\mathbf{P}^{(1)} = \mathbf{I} \tag{3.118}$$

and subsequent $\mathbf{P}^{(k)}$ are updated as

$$\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} - \frac{\mathbf{P}^{(k)}\gamma^{(k)}\gamma^{(k)T}\mathbf{P}^{(k)}}{\gamma^{(k)T}\mathbf{P}^{(k)}\gamma^{(k)}}. \tag{3.119}$$

Again this method is equivalent to other described methods for quadratic functions. When applied to general functions, $\mathbf{P}^{(i)}$ must be reset to $\mathbf{I}$ every n iterations since $\mathbf{P}^{(n+1)} = 0$. The method has the descent property $\mathbf{s}^{(k)T}\mathbf{g}^{(k)} \leq 0$, but has a disadvantage that matrix calculations are required in each iteration.

3.6.2 Direction Set Methods

Direction set methods[1],[26] generate conjugate directions with respect to the Hessian matrix G, when they are applied to a quadratic function, without use of derivative information. Construction of the conjugate direction is based on the following theorem[1]:

Theorem 3.6: Parallel subspace property

Let us have a quadratic function $q(\mathbf{x})$ with a positive definite Hessian matrix $\mathbf{G}$. Consider two parallel subspaces S_1 and S_2, generated by independent directions $\mathbf{s}^{(1)}, ..., \mathbf{s}^{(k)}$ ($k < n$) from the points $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$, such that $S_1 \neq S_2$, i.e.

$$S_1 = \left\{ \mathbf{x};\, \mathbf{x} = \mathbf{v}^{(1)} + \sum_{i=1}^{k} \alpha_i \mathbf{s}^{(i)} \ \forall \alpha_i \right\}$$

and similarly S_2. Let $\mathbf{z}^{(1)}$ be the point which minimises q on S_1 and $\mathbf{z}^{(2)}$ the point which minimises q on S_2 Then $\mathbf{z}^{(2)}-\mathbf{z}^{(1)}$ is conjugate to $\mathbf{s}^{(1)},...,\mathbf{s}^{(k)}$ with respect to $\mathbf{G}$, i.e. $\left(\mathbf{z}^{(2)}-\mathbf{z}^{(1)}\right)^T\mathbf{G}\,\mathbf{s}^{(i)}=0,\ i=1,...,k$.

The above theorem is outlined for two dimensions in Figure 3.8. Although instructive, a two-dimensional representation is not completely satisfactory because the complement of the vector space spanned by $\mathbf{s}^{(1)}$ is one dimensional and therefore the line $\mathbf{z}^{(1)}+\alpha\left(\mathbf{z}^{(2)}-\mathbf{z}^{(1)}\right)$ contains a minimum of $q(x_i,x_2)$. Also the parallel subspaces are only one dimensional.

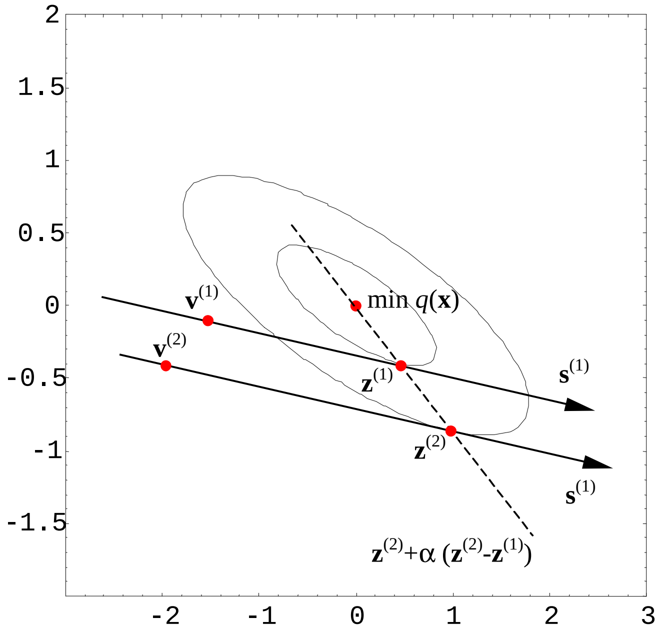

Figure 3.8: The parallel subspace property in two dimensions.

In the direction set methods, conjugate directions $\mathbf{s}^{(i)}$ are generated one by one by minimising the function in a subspace spanned by previously constructed $\mathbf{s}^{(1)},...,\mathbf{s}^{(i-1)}$ (giving $\mathbf{x}^{(i)}$, say), a parallel subspace is created by displacing the obtained minimum by a vector that is independent on $\mathbf{s}^{(1)},...,\mathbf{s}^{(i-1)}$ ($\mathbf{d}^{(i)}$, say), followed by minimising in that subspace (giving $\mathbf{z}^{(i)}$, say), and setting a new conjugate direction to $\mathbf{s}^{(i)}=\mathbf{z}^{(i)}-\mathbf{x}^{(i)}$. Since the directions $\mathbf{s}^{(1)},...,\mathbf{s}^{(i)}$ are conjugate, previously performed minimisation in directions $\mathbf{s}^{(1)},...,\mathbf{s}^{(i-1)}$ is not affected by moving along the line $\mathbf{z}^{(i)}+\alpha\left(\mathbf{z}^{(i)}-\mathbf{x}^{(i)}\right)$. Minimisation in the subspace

$$S_1^{(i)} = \left\{ \mathbf{x};\, \mathbf{x} = \mathbf{z}^{(i)} + \sum_{j=1}^{i} \alpha_j \mathbf{s}^{(j)} \;\; \forall \alpha_j \right\}$$

is therefore achieved by line minimisation along that line (this is well illustrated in Figure 3.8). This is of course valid only for quadratic functions, but the construction of an algorithm for general functions is based on the same arguments and is similar.

From the above construction it is clear that the direction $\mathbf{s}^{(i)}$ is a linear combination of vectors $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, ..., \mathbf{d}^{(i)}$, but is not contained in the subspace spanned by $\mathbf{d}^{(i+1)}, \mathbf{d}^{(i+2)}, ..., \mathbf{d}^{(n)}$

The above described procedure is followed in Smith's method. The cycle is repeated when the function is not quadratic. The method requires independent vectors $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, ..., \mathbf{d}^{(n)}$ to be supplied, which are used for the successive generation of parallel subspaces after minimisation in the current subspace. This is however not the most effective approach because directions are not treated equally. Line minimisations are performed more frequently in the directions that are constructed earlier.

This deficiency is abolished in Powell's method. Its cycle is as described above, except that a point of the parallel subspace is obtained by line minimisations along $\mathbf{d}^{(i)}, \mathbf{d}^{(i+1)}, ..., \mathbf{d}^{(n)}$ from $\mathbf{x}^{(i)}$ rather by just adding $\mathbf{d}^{(i)}$ to $\mathbf{x}^{(i)}$. Powell's method is also such that cycles can be continued when the function is not quadratic, while in Smith's method directions are restarted after each cycle. The algorithm is sketched below.

Algorithm 3.8: Powell's direction set algorithm.

Given independent directions $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, ..., \mathbf{s}^{(n)}$, the minimum point $\mathbf{x}^{(1)}$ along $\mathbf{s}^{(1)}$ is found by a line search. Then the following iteration is repeated for $i = 1, 2, ...$:

1. Find

$$\mathbf{z}^{(i)} = \mathbf{x}^{(i)} + \sum_{j=1}^{n} \alpha_j \mathbf{s}^{(j)}$$

by sequentially minimising f along $\mathbf{s}^{(1)},\ \mathbf{s}^{(2)},\ \ldots,\ \mathbf{s}^{(n)}$. If $i \le n$ then the last i directions have already been replaced by conjugate directions.

2. For $j = 1, 2, ..., n-1$ replace $\mathbf{s}^{(j)}$ by $\mathbf{s}^{(j+1)}$.

3. Set $\mathbf{s}^{(n)} = \mathbf{z}^{(i)} - \mathbf{x}^{(i)}$, which is a new conjugate direction. Find the minimum point $\mathbf{x}^{(i+1)} = \mathbf{z}^{(i)} + \alpha \mathbf{s}^{(n)}$ along $\mathbf{s}^{(n)}$ by a line search. For a quadratic function, when $i \le n$, $\mathbf{x}^{(i+1)}$ is a minimum point of the subspace

$$S = \left\{ \mathbf{x}; \mathbf{x} = \mathbf{z}^{(i)} + \sum_{k=n-i}^{n} \alpha_k \mathbf{s}^{(k)} \right\}.$$

since the last $i+1$ directions are conjugate.

The first *n-i* line searches in line 1 locate a point in a parallel subspace. The last *i* line searches can be thought of as minimisation of that subspace spanned by the last *i* directions (which are conjugate).

The algorithm terminates in about n^2 line searches when applied to quadratic functions. This is about twice as much as Smith's method ($\frac{1}{2}n(n+1)$), but the directions are now treated equally. Pseudo-conjugate directions are retained for a general function after the *n*-th iteration and are updated from iteration to iteration. The method is therefore more effective for general functions as Smith's method. One of the problems with this method is that in some problems directions $\mathbf{s}^{(j)}$ tend to become linearly dependent. It is possible to introduce modifications which deal with this problem. One possibility is to reset the direction set every certain numbers of the cycles.

3.7 Restricted Step Methods

The restricted step prototype algorithm is an alternative to the line search strategy, in an attempt to ensure global convergence of minimisation algorithms.

There is one fundamental difference between the line search and restricted step approach. As can be seen from precedent sections, the line search based algorithms rely to a great extent on a quadratic model. Directions of line searches are essentially constructed in such a way that when an algorithm is applied to a quadratic function with a positive definite Hessian matrix, termination occurs in a finite number of exact line searches. Then, using the argument that every sufficiently smooth function can be locally (i.e. in the neighbourhood of the current guess) approximated by a quadratic function, the strategy designed on a quadratic model is more or less directly transferred to algorithms for handling general functions.

The fact that even within a single line search the minimised function can deviate far from its quadratic approximation is ignored by the line search approach. On the contrary, the main idea of the restricted step approach is to make direct use of the quadratic model, but only in the limited region where this is an adequate approximation to the original function. This leads to a sub-problem of minimsation of a quadratic approximation, limited to a certain region. One of the benefits is that difficulties with a non-positive definite Hessian matrix are avoided. It is clear at first sight that among the important concerns of restricted step methods is how to define a so called trust region $\mathbf{\Omega}^{(k)}$, i.e. the neighbourhood of a current guess in which the use of a quadratic approximation is adequate.

In the view of the above discussion, consider the problem

$$\begin{aligned} &\textit{minimise} && q^{(k)}(\delta) \\ &\textit{subject to} && \|\delta\| \le h^{(k)}, \end{aligned} \tag{3.120}$$

where $q^{(k)}(\delta)$ is a second order Taylor approximation of f about $\mathbf{x}^{(k)}$ and $\delta = \mathbf{x} - \mathbf{x}^{(k)}$, i.e.

$$q^{(k)}(\delta) = f^{(k)} + \mathbf{g}^{(k)T}\delta + \tfrac{1}{2}\delta^T \mathbf{G}^{(k)}\delta . \tag{3.121}$$

The second part of equation (3.120) defines the trust region as

$$\Omega^{(k)} = \left\{\mathbf{x}; \left\|\mathbf{x} - \mathbf{x}^{(k)}\right\| \le h^{(k)}\right\}. \tag{3.122}$$

Restricted step methods aim at keeping the step restriction $h^{(k)}$ as large as possible, subject to the restriction that a certain agreement between $q^{(k)}\left(\delta^{(k)}\right)$ and $f\left(\mathbf{x}^{(k)} + \delta^{(k)}\right)$ must be retained, where $\delta^{(k)}$ is the solution of (3.120). Some measure of agreement must be defined for this purpose. Let us define the actual reduction of f in the step k as

$$\Delta f^{(k)} = f^{(k)} - f\left(\mathbf{x}^{(k)} + \delta^{(k)}\right) \tag{3.123}$$

and the corresponding predicted reduction as

$$\Delta q^{(k)} = q^{(k)}(0) - q^{(k)}\left(\delta^{(k)}\right) = f^{(k)} - q^{(k)}\left(\delta^{(k)}\right). \tag{3.124}$$

Then the ratio

$$r^{(k)} = \frac{\Delta f^{(k)}}{\Delta q^{(k)}} \tag{3.125}$$

is a suitable measure of the agreement, where good agreement is indicated by $r^{(k)}$ being close to 1. In restricted step algorithms it is attempted to maintain a certain degree of agreement by adaptively changing $h^{(k)}$. Such a prototype algorithm is defined below.

Algorithm 3.9: Iteration of the restricted step prototype algorithm.

1. Given $\mathbf{x}^{(k)}$ and $h^{(k)}$, evaluate $f^{(k)}$, $\mathbf{g}^{(k)}$ and $\mathbf{G}^{(k)}$ to define $q^{(k)}(\delta)$ in (3.121).
2. Solve (3.120) for $\delta^{(k)}$.
3. Evaluate $f\left(\mathbf{x}^{(k)} + \delta^{(k)}\right)$ and $r^{(k)}$.
4. If $r^{(k)} < \tau_1$ set $h^{(k+1)} = \rho_1 \left\|\delta^{(k)}\right\|$,
   if $r^{(k)} > \tau_2$ and $\left\|\delta^{(k)}\right\| = h^{(k)}$ set $h^{(k+1)} = \rho_2 h^{(k)}$,
   else set $h^{(k+1)} = h^{(k)}$.
5. If $r^{(k)} \le 0$ set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$ else set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}$.

Constants used in the above algorithm must be chosen so that $0 < \tau_1 < \tau_2 < 1$, $0 < \rho_1 < 1$ and $1 < \rho_2$. A suitable choice is $\tau_1 = 0.25$, $\tau_2 = 0.75$, $\rho_1 = 0.25$ and $\rho_2 = 2$, but the algorithm is not very sensitive to the choice[1]. In line 4 of the algorithm the step restriction is tightened if agreement between $\Delta f^{(k)}$ and $\Delta q^{(k)}$ is bad, and relaxed if the agreement is good and at the same time the minimum of q lies on the edge of the trust region. If the minimum of $q^{(k)}$ lies inside the trust region, then there is no need to further relax the step restriction because that constraint will become inactive anyway, and the algorithm will reduce to the basic Newton method with rapid convergence. In line 5 $\mathbf{x}^{(k)}$ is preserved if $f\left(\mathbf{x}^{(k)} + \delta^{(k)}\right) > f^{(k)}$, which guarantees the descent property $f^{(k+1)} \le f^{(k)}$.

The following strong convergence result holds for restricted step methods[1]:

Theorem 3.7:

For Algorithm 3.9, if $\mathbf{x}^{(k)} \in B \subset \mathbb{R}^n \ \forall k$, where B is bounded, and if $f \in \mathbb{C}^2$ on B, then there exists an accumulation point $\mathbf{x}^\infty$ which satisfies the first and the second order necessary conditions for a local minimum. If $\mathbf{G}$ also satisfies the Lipshitz condition $\left\|\mathbf{G}(\mathbf{x})-\mathbf{G}(\mathbf{y})\right\| \leq \lambda\left\|\mathbf{x}-\mathbf{y}\right\|$ in some neighbourhood of $\mathbf{x}^{\infty}$ and if $\mathbf{G}^{\infty}$ is positive definite, then for the main sequence $r^{(k)} \rightarrow 1$, $\inf h^{(k)} > 0$, the constraint $\left\|\delta\right\| \leq h^{(k)}$ is inactive for sufficiently large k, and convergence is second order.

The existence of B in the above theorem is not a strong requirement. It is implied if any level set $\left\{\mathbf{x};\, f\left(\mathbf{x} \leq f^{(k)}\right)\right\}$ is bounded because $\mathbf{x}^{(k)}$ are descent.

If the L_{∞} norm is used in the step restriction, then the subproblem (3.120) becomes a quadratic programming problem with simple bounds, for which good algorithms for local solutions exist. It is however difficult to find global solutions, but in practice it is adequate to calculate only local solutions.

Slow progress of the method can arise if the norm is not scaled. Ideally the natural metric norm $\left\|\delta\right\|_{G} = \delta^{T}\mathbf{G}\delta$ would be chosen when **G** is positive definite, but the scaling of variables can also be an adequate approach.

The Hessian matrix in the restricted step methods can be replaced by the approximate Hessian $\mathbf{B}^{(k)}$ or its inverse $\mathbf{H}^{(k)}$, updated according to a quasi-Newton scheme. In such a case similar global convergence result holds as for the original method.

3.8 Basics of Constrained Optimisation

The remainder of chapter 3 is devoted to the case in which constraints on the optimisation variables are defined (Figure 3.10). The presence of constraints introduces additional complexity in the treatment of local solutions in view of the definition of necessary and sufficient conditions, which is discussed in the present section.

Constrained optimisation problems are much more difficult to treat numerically than unconstrained problems. Many algorithms for their solution are based on transformation of the constrained problem to a sequence of unconstrained optimisation subproblems whose solutions converge to the solution of the constrained problem. A commonly used approach is the penalty function approach based on addition of weighted penalty terms to the objective function which cause high values where constraints are violated or close to be violated. In the limit when weights tend to infinity, the solutions of the unconstrained problems tend to the solution of the constrained problem. This approach is described in section 3.10.

The next important approach is the elimination of variables. Equality constraints are used to define implicit dependence (through solution of a nonlinear systems of equations defined by equality constraints) of a subset of optimisation variables on the remaining variables. The constrained problem is in this way transformed to an unconstrained problem defined on a reduced set of variables, but each evaluation requires a system of nonlinear equations to be solved for the dependent variables. When inequality constraints are involved, the active constraints are treated as equality constraints. Since it is not known in advance which constraints are active in the solution, the set of active constraints is iteratively updated. This leads to the active set type of methods, the principle of which are described in sections 3.9 and 3.11.

Algorithms for solution of constrained optimisation problems are based on quadratic models to a large extent. Some algorithms for general functions explicitly generate quadratic programming subproblems (quadratic objective function and linear constraints). These algorithms represent an alternative to the more traditional penalty function approach and seem to be superior from the point of view of efficiency. Section 3.9 covers some basic aspects of quadratic programming.

There are also solution algorithms which linearise both the objective functions and constraints about the current iterate and therefore generate a sequence of linear programming problems[14][9]. This approach seems to be popular in some fields, however only problems with some special structure can be successfully solved in this way (e.g. with the objective function close to linear), therefore attention is not devoted to the approach in this work. Linear programming (linear objective function and constraints) is also not treated in this work for the same reason.

3.8.1 Langrange Multipliers and First Order Conditions for Unconstrained Local Minima

Consider the problem (3.19) where constraints are present. For any point $\mathbf{x}'$ active or binding constraints are those for which the corresponding constraint function is zero at that point. A set of their indices will be denoted by

$$\mathcal{A}' = \mathcal{A}(\mathbf{x}') = \{ i; c_i(\mathbf{x}') = 0 \} \qquad (3.126)$$

Any constraint is active at $\mathbf{x}'$ if that point is on the boundary of its feasible region. The set of active constraints at the solution $\mathcal{A}^*$ is of particular importance.

Constraints which are not active at the solution can be perturbed by small amounts without affecting the problem solution.

The gradient of the *i*-th constraint function ∇c_i will be denoted by $\mathbf{a}_i$ and referred to as the normal vector of the constraint c_i. These vectors can be arranged in a Jacobian matrix **A**, whose columns are constraint gradients.

Consider a problem with only equality constraints and a feasible incremental step δ taken from a local minimiser. By a Taylor series we have

$$c_i\left(\mathbf{x}^* + \delta\right) = c_i^* + \delta^T \mathbf{a}_i^* + o\left(\|\delta\|\right).$$

Since δ is a feasible step we have $c_i\left(\mathbf{x}^* + \delta\right) = c_i^* = 0$ and where the length of the step length is small, we have by neglecting higher order terms $\delta^T \mathbf{a}_i^* = 0$. By taking into account all constraints, we can define a feasible direction as a direction which satisfies

$$\mathbf{s}^T \mathbf{a}_i^* = 0 \quad \forall i \in E \ . \tag{3.127}$$

Clearly if **s** is a feasible direction then –**s** is also a feasible direction. Since $\mathbf{x}^*$ is a constrained local minimiser, there is no feasible descent direction, because otherwise *f* could be reduced by an arbitrarily small step in that direction. It follows that $\mathbf{s}^T \mathbf{g}^* = 0$ for any feasible direction **s**. Due to (3.127) this is satisfied if $\mathbf{g}^*$ is a linear combination of constraint gradients, i.e.

$$\mathbf{g}^* = \sum_{i \in E} \mathbf{a}_i^* \lambda_i^* = \mathbf{A}^* \lambda^* \ . \tag{3.128}$$

Multipliers λ_i^* are referred to as Lagrange multipliers and can be arranged in the Lagrange multiplier vector (denoted by λ^* without a subscript). The above equation is also a necessary condition for a local minimiser. If (3.128) would not hold, then $\mathbf{g}^*$ could be expressed as

$$\mathbf{g}^* = \mathbf{A}^* \lambda^* + \mu \tag{3.129}$$

where μ is a component of $\mathbf{g}^*$ orthogonal to all $\mathbf{a}_i^*$. Then $\mathbf{s} = -\mu$ would be a feasible descent direction (i.e. would satisfy both (3.128) and $\mathbf{s}^T \mathbf{g}^* < 0$). A feasible incremental step δ along $\mathbf{s}$ would reduce f, which contradicts the fact that $\mathbf{x}^*$ is a local minimiser. This is illustrated in Figure 3.9 for a single constraint.

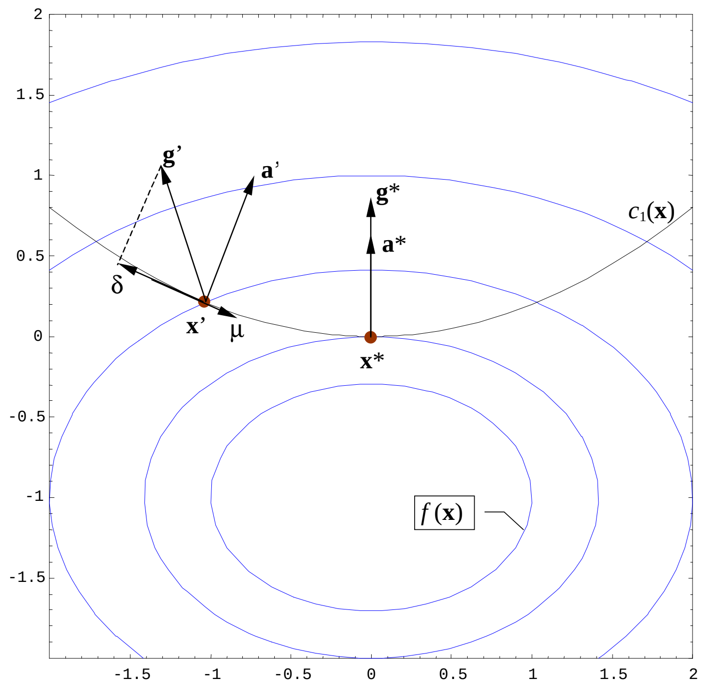

Figure 3.9: Illustration of necessary conditions for a constrained local minimum.

The necessary conditions are a basis of the method of Lagrange multipliers for equality constraint problems. The method searches for vectors $\mathbf{x}^*$ and λ^*, which solve the equations

$$\mathbf{g}(\mathbf{x}) = \sum_{i \in E} \lambda_i \mathbf{a}_i(\mathbf{x})$$

and (3.130)

$$c_i(\mathbf{x}) = 0,\; i \in E .$$

This approach has a similar disadvantage to the Newton method for unconstrained minimisation: the above equations are satisfied in a constrained saddle point or maximiser, since no second order information is taken into account.

The above equations can be written in a simpler form if we define the Lagrangian function

$$\mathcal{L}(\mathbf{x},\lambda) = f(\mathbf{x}) - \sum_i \lambda_i c_i(\mathbf{x}). \tag{3.131}$$

Equations (3.130) them become

$$\overline{\nabla}\mathcal{L}(\mathbf{x},\lambda) = 0, \tag{3.132}$$

where

$$\overline{\nabla} = \begin{bmatrix} \nabla_x \\ \nabla_\lambda \end{bmatrix} \tag{3.133}$$

is the gradient operator in the $n+m$ dimensional variable space (m will denote the number of constraints). We see that a necessary condition for a local minimiser is that $(\mathbf{x}^*, \lambda^*)^T$ is a stationary point of the Lagrangian function.

Lagrange multipliers have a clear practical interpretation. If the Jacobian matrix of constraints has rank m (linearly independent $\mathbf{a}_i$) then the multipliers in (3.128) are uniquely defined by

$$\lambda^* = \mathbf{A}^{*+}\mathbf{g}, \tag{3.134}$$

where $\mathbf{A}^{*+} = \left(\mathbf{A}^{*T}\mathbf{A}^*\right)^{-1}\mathbf{A}^T$ is a generalised inverse[2] of $\mathbf{A}^*$. Consider in such case perturbations of the right-hand sides of the constraint

$$c_i(\mathbf{x}) = \varepsilon_i,\ i \in E \tag{3.135}$$

and let $f(\varepsilon)$ and $\lambda(\varepsilon)$ denote how the solution and multipliers change with respect to perturbations. The lagrangian function for the perturbed problems is

$$\mathcal{L}(\mathbf{x},\lambda,\varepsilon) = f(\mathbf{x}) - \sum_{i \in E} \lambda_i \left(c_i(\mathbf{x}) - \varepsilon_i\right) \tag{3.136}$$

In the perturbed solution new constraints are satisfied, therefore

$$f(\mathbf{x}(\varepsilon)) = \mathcal{L}(\mathbf{x}(\varepsilon), \lambda(\varepsilon), \varepsilon).$$

Derivation of this equation with respect to ε_i gives

$$\frac{d\ f}{d\,\varepsilon_i} = \frac{d\mathcal{L}}{d\varepsilon_i} = \frac{\partial \mathbf{x}^T}{\partial \varepsilon_i}\nabla_x \mathcal{L} + \frac{\partial \lambda^T}{\partial \varepsilon_i}\nabla_\lambda \mathcal{L} + \frac{\partial \mathcal{L}}{\partial \varepsilon_i}$$

By (3.132), (3.135) and (3.136) it follows that

$$\frac{df}{d\varepsilon_i} = \lambda_i\,. \tag{3.137}$$

Lagrange multipliers therefore indicate how sensitive the value of the objective function at the solution is to changes in the corresponding constraints.

Consider now a case where inequality constraints are present. Only active constraints at the solution $\mathcal{A}^*$ influence conditions for the solution. A set of active inequality constraints at the solution will be denoted by $I^*\ (= \mathcal{A}^* \cap I)$. Any feasible direction **s** must satisfy (in addition to (3.127)) the condition

$$\mathbf{s}^T \mathbf{a}_i^* \geq 0 \quad \forall i \in I^*\,. \tag{3.138}$$

Conditions for a local minimiser are

$$\mathbf{g}^* = \sum_{i \in \mathcal{A}^*} \lambda_i^* \mathbf{a}_i^* \tag{3.139}$$

and

$$\lambda_i^* \geq 0 \ \ \forall i \in I^*\,. \tag{3.140}$$

Condition (3.139) can be deduced in a similar way to (3.130). Condition (3.140), which is an extra condition with respect to the case of equality constraints, can be deduced using the result (3.137). A small perturbation of the *i*- th active inequality constraint by positive ε_i induces a change $\mathbf{x}(\varepsilon)$ that is feasible with respect to the unperturbed problem. Therefore *f* must not decrease, which implies $df^* / d\varepsilon_i \geq 0$ and hence $\lambda_i^* \geq 0$.

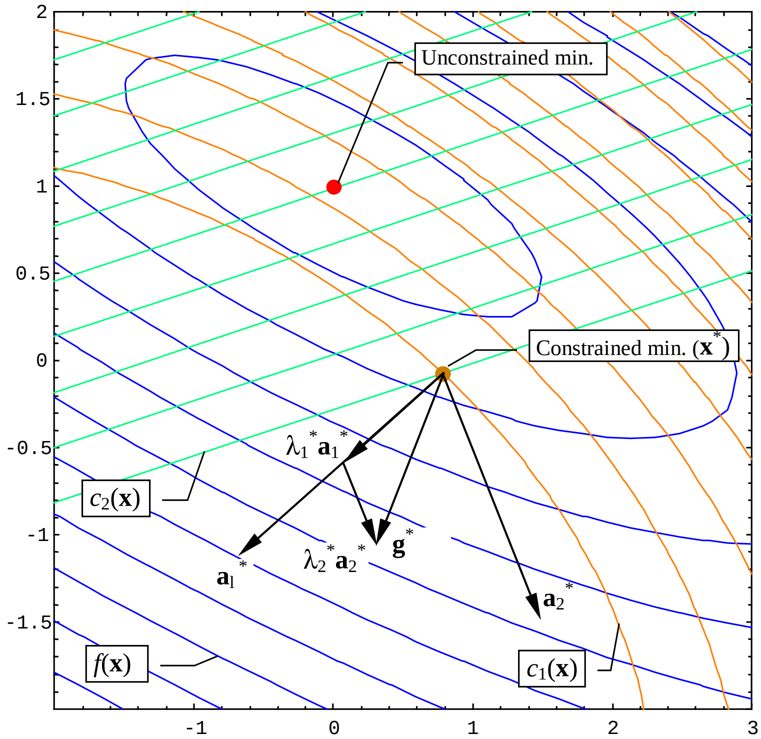

Figure 3.10: Constrained optimisation problem with two inequality constraints. Contours of constraint functions are drawn only in the infeasible region where their values are less or equal to zero. Since both constraints are active in the solution, the solution would remain unchanged if one or both constraints were replaced by equality constraints.

Lagrange multipliers have another important interpretation in the case of inequality constraints. Consider a point at which (3.139) is satisfied and (3.140) holds for all i except for $i = p$, i.e. $\lambda_p^* < 0$, and let all $\mathbf{a}_i^*$ be linearly independent. Then it is possible to find a direction $\mathbf{s}$ for which $\mathbf{s}^T \mathbf{a}_p^* = 1$ and $\mathbf{s}^T \mathbf{a}_i^* = 0$ for all other active constraints. Such is given for example by $\mathbf{s} = \mathbf{A}^{*+T} \mathbf{e}_p$, where $\mathbf{e}_p$ is the p-th coordinate vector. Then $\mathbf{s}$ is a feasible direction and at the same time a descent direction since

$$\mathbf{s}^T \mathbf{g}^* = \mathbf{s}^T \mathbf{a}_p^* \lambda_p^* < 0. \tag{3.141}$$

This first contradicts the fact that $\mathbf{x}^*$ is a local minimiser and is another proof that conditions (3.140) are necessary. Besides, it indicates that $f(\mathbf{x})$ can be reduced by moving away from the boundary of constraint p for which the corresponding Lagrange multiplier is negative. This is important in the active set methods for handling inequality constraints, where constraints index p with $\lambda_p^* < 0$ can be removed from the active set (section 3.9.2).

In the derivation of the first order conditions the regularity assumption that $\mathbf{a}_i^*$ are independent was made. This is not necessarily the case and an exact statement of the conditions requires more careful treatment[1].

First the notion of feasible direction must be defined more exactly. Consider a feasible point $\mathbf{x}'$ and any infinite sequence of feasible points convergent to this point $\left\{\mathbf{x}^{(k)}\right\} \to \mathbf{x}'$ where in addition $\mathbf{x}^{(k)} \neq \mathbf{x}'$ for all k. It is possible to write

$$\mathbf{x}^{(k)} - \mathbf{x}' = \delta^{(k)} \mathbf{s}^{(k)} \quad \forall k \tag{3.142}$$

where $\delta^{(k)} > 0$ are scalars and $\mathbf{s}^{(k)}$ are vectors of any fixed length $\sigma > 0$. A directional sequence is defined as any such sequence for which vectors $\mathbf{s}^{(k)}$ converge to some direction, i.e. $\mathbf{s}^{(k)} \to \mathbf{s}$. The limiting vector $\mathbf{s}$ is then referred to as the feasible direction. $\mathcal{F}(\mathbf{x}') = \mathcal{F}'$ will be used to denote the set of all feasible directions at $\mathbf{x}'$.

It can be seen from the previous discussion that the set of feasible directions for the linearised constraint set is

$$F(\mathbf{x}') = F' = \left\{ \mathbf{s};\, \mathbf{s} \neq 0 \wedge \begin{array}{ll} \mathbf{s}^T \mathbf{a}_i' = 0 & \forall i \in E \\ \mathbf{s}^T \mathbf{a}_i' \geq 0 & \forall i \in I' \end{array} \right\}, \tag{3.143}$$

where I' is a set of active inequality constraints at $\mathbf{x}'$.

The relation $\mathcal{F}' \subseteq F'$ holds in general. $\mathcal{F}' = F'$ either if the constraints $i \in \mathcal{A}'$ are linear or vectors $\mathbf{a}_i'$, $i \in \mathcal{A}'$ are linearly independent. The assumption $\mathcal{F}' = F'$ is referred to as a constraint qualification at $\mathbf{x}'$.

The set of descent directions at $\mathbf{x}'$ is defined as

$$\mathcal{D}(\mathbf{x}') = \mathcal{D}' = \left\{ \mathbf{s};\, \mathbf{s}^T \mathbf{g}' < 0 \right\}. \tag{3.144}$$

[1] In some optimisation literature the possibility that gradients of active constraints in the solution can be linearly dependent is ignored, sometimes with an argument that this is an extremely unlikely situation.

If $\mathbf{x}^*$ is a local minimiser, then $\mathcal{F}^* \cap \mathcal{D}^* = \emptyset$, i.e. no feasible descent directions exist.

Let the following regularity assumption be made:

$$F^* \cap \mathcal{D}^* = \mathcal{F}^* \cap \mathcal{D}^*. \tag{3.145}$$

This is weaker assumption than $\mathcal{F}' \subseteq F'$. Under this assumption the following more general statement of the first order necessary conditions can be made[1],[4],[7]:

Theorem 3.8: Kuhn-Tucker (or KT) conditins.

If $\mathbf{x}^*$ is a local constrained minimiser and if regularity assumption(3.145) holds, then there exist Lagrange multipliers λ^* such that $\mathbf{x}^*$ and λ^* satisfy the following system:

$$\begin{aligned}
&\nabla_x \mathcal{L}(\mathbf{x},\lambda) = 0\\
&c_i(\mathbf{x}) = 0, \quad i \in E\\
&c_i(\mathbf{x}) \geq 0, \; i \in I\\
&\lambda_i \geq 0, \; i \in I\\
&\lambda_i c_i(\mathbf{x}) = 0 \;\; \forall i.
\end{aligned} \tag{3.146}$$

A point that satisfies the above conditions is referred to as a KT point. The condition $\lambda_i^* c_i^* = 0$ is referred to as the complementarity condition. It states that both λ_i^* and c_i^* can not be non-zero, which means that inactive constraints are regarded as having zero Lagrange multipliers. If there is no *i* such that $\lambda_i^* = c_i^* = 0$ then strict complementarity is said to hold. The case $\lambda_i^* = c_i^* = 0$ appears for example if an unconstrained minimiser lies on the boundary of the feasible region, which is an intermediate state between a constraint being strongly active and inactive.

3.8.2 Second Order Conditions

Consider first the case with only equality constraints. The second order conditions can be derived from the second order Taylor series of the Lagrangian function about the local solution. It is assumed that $\mathbf{a}_i^*$ are independent so that unique Lagrange multipliers exist. Let a feasible incremental step δ be made along any feasible direction **s**. By feasibility it follows that $f(\mathbf{x}+\delta)=\mathcal{L}(\mathbf{x}+\delta,\lambda)$. We also take into account that $\mathcal{L}$ is stationary at $\mathbf{x}^*$ and λ^* to eliminate the first derivatives. The second order Taylor expansion then gives (after neglecting higher than second order terms)

$$\begin{aligned} &f(\mathbf{x}^*+\delta)=\mathcal{L}(\mathbf{x}^*+\delta,\lambda^*)\approx \\ &\mathcal{L}(\mathbf{x}^*+\delta,\lambda^*)+\delta^T\underbrace{\nabla_x\mathcal{L}(\mathbf{x}^*,\lambda^*)}_{=0}+\tfrac{1}{2}\delta^T\mathbf{W}\delta=. \\ &f^*+\tfrac{1}{2}\delta^T\mathbf{W}\delta \end{aligned} \tag{3.147}$$

W denotes the Hessian matrix of the Lagrangian function with respect to variables **x**:

$$\mathbf{W}^*=\nabla_x{}^2\mathcal{L}(\mathbf{x}^*,\lambda^*)=\nabla^2 f(\mathbf{x}^*)-\sum_i\lambda_i^*\nabla^2 c_i(\mathbf{x}^*). \tag{3.148}$$

Since $\mathbf{x}^*$ is a local minimiser, the function value taken in any feasible infinitesimal incremental step in any direction must be greater than or equal to f^*. It follows that

$$\mathbf{s}^T\mathbf{W}\mathbf{s}^*\geq 0 \tag{3.149}$$

for any feasible direction, i.e. for any **s** that satisfies

$$\mathbf{a}_i^{*T}\mathbf{s}=0\ \ \forall i\in E. \tag{3.150}$$

This is a second order necessary condition for a local minimiser, which can also be stated as a requirement that the Lagrangian function must have a non-negative curvature along any feasible direction.

A sufficient condition is that $\mathbf{x}^*$ satisfies (3.128) and

$$\mathbf{s}^T\mathbf{W}^*\mathbf{s}>0 \tag{3.151}$$

for all feasible directions **s** that satisfy (3.150) (a zero vector is not considered to be a feasible direction).

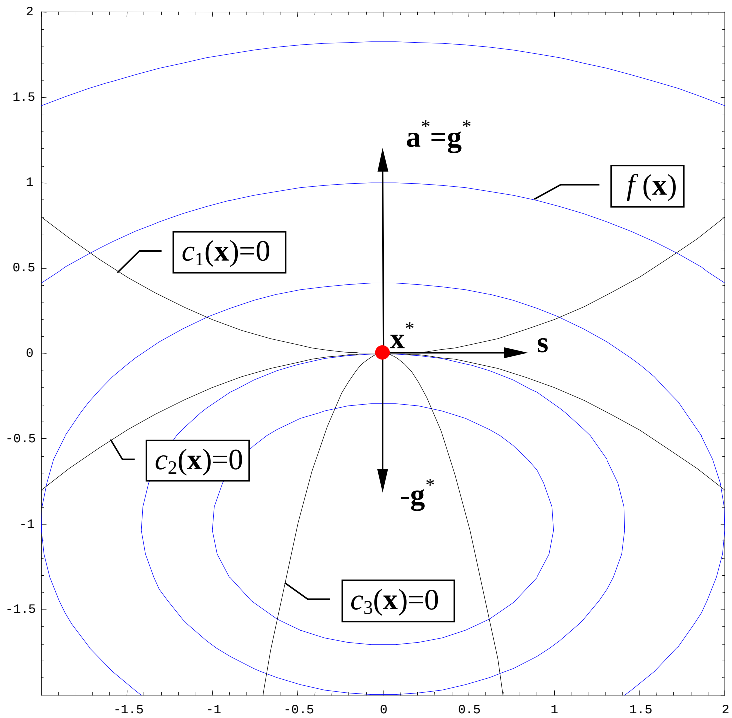

Figure 3.11: Illustration of the second order conditions. A problem with three different equality constraints is shown. In all three cases $\mathbf{a}^* = \mathbf{g}^*$ and $\lambda^* = 1$. The problem with constraint function c_3 does not match the necessary conditions for $\mathbf{x}^*$ to be a local minimiser because $\mathbf{s}^T\left(\nabla^2 f\left(\mathbf{x}^*\right)\right)\mathbf{s} < \mathbf{s}^T\left(\lambda\nabla^2 c_3\left(\mathbf{x}^*\right)\right)\mathbf{s}$ and thus $\mathbf{s}^T \mathbf{W}_3^* \mathbf{s} < 0$. The second constraint satisfies necessary but not sufficient second order conditions and therefore higher order terms of the Taylor series become significant.

When inequality constraints are present, again only active constraints affect matters. We can also realize that the second order conditions are important only along feasible stationary directions ($\mathbf{s}^T\mathbf{g}^* = 0$ with respect to constraints), but not along ascent directions where first order information is sufficient. If an inequality constraint $c_i(\mathbf{x}) \geq 0$ is present with $\lambda_i^* > 0$, then directions for which $\mathbf{s}^T\mathbf{a}_i^* > 0$ are ascent directions (according to the discussion regarding (3.141)). Stationary directions satisfy

$$\mathbf{s}^T\mathbf{a}_i^* = 0 \quad \forall i \in \mathcal{A}^* \tag{3.152}$$

The second order necessary conditions are then that (3.149) holds for all **s** that satisfy (3.152). Sufficient conditions for a strict local minimiser are that the Kuhn-Tucker conditions with strict complementarity ($\lambda_i^* > 0 \; \forall i \in I^*$) hold and

$$\mathbf{s}^T \mathbf{W}^* \mathbf{s} > 0 \quad \forall \mathbf{s} : \mathbf{s}^T \mathbf{a}_i^* = 0, i \in \mathcal{A}^* . \tag{3.153}$$

Exact statement of these conditions[1] again relies on some regularity assumption. Let us define a set of strictly active constraints

$$\mathcal{A}_+^* = \left\{ i; i \in E \vee \lambda_i^* > 0 \right\}. \tag{3.154}$$

Consider feasible directional sequences for which $\mathbf{x}^{(k)} \to \mathbf{x}^*$ for which

$$c_i\left(\mathbf{x}^{(k)}\right) = 0 \;\; \forall i \in \mathcal{A}_+^* . \tag{3.155}$$

and define $\mathcal{G}^*$ as a set of all resulting feasible directions. A corresponding set where constraints which determine $\mathcal{G}^*$ are linearised can then be defined as

$$G^* = \left\{ \mathbf{s}; \mathbf{s} \neq 0 \wedge \begin{array}{c} \mathbf{a}_i^{*T} \mathbf{s} = 0, i \in \mathcal{A}_+^* \\ \mathbf{a}_i^{*T} \mathbf{s} \geq 0, i \in \mathcal{A}^* \setminus \mathcal{A}_+^* \end{array} \right\}. \tag{3.156}$$

$\mathcal{G}^* \subseteq G^*$ holds and in order to state the second order necessary conditions, the regularity assumption

$$\mathcal{G}^* = G^* \tag{3.157}$$

is made. The second order necessary and sufficient conditions can then be stated as below[1],[4]:

Theorem 3.9 (second order necessary conditions):

If $\mathbf{x}^*$ is a constrained local minimiser and if the regularity assumption (3.145) holds, then there exist multipliers λ^* such that Theorem 3.8 is valid (i.e. $\mathbf{x}^*$ is a KT point). For any such λ^*, if also the regularity assumption (3.157) holds, it follows that

$$\mathbf{s}^T \mathbf{W}^* \mathbf{s} \geq 0 \;\; \forall \mathbf{s} \in \mathbf{G}^* . \tag{3.158}$$

Theorem 3.10 (second order sufficient conditions):

If at $\mathbf{x}^*$ there exist multipliers λ^* such that conditions (3.146) hold, and if

$$\mathbf{s}^T \mathbf{W}^* \mathbf{s} > 0 \quad \forall \mathbf{s} \in G^*, \tag{3.159}$$

then $\mathbf{x}^*$ is a strict local minimiser.

3.8.3 Convex Programming Results

Some strong theoretical results hold when the objective function is a convex function and when the feasible region is a convex set. Within the scope of this work these results are not important because of direct applicability to specific problems, but are important for treatment of subproblems that arise in some optimisation algorithms. Convex programming result are also important for statement of the duality principles, which are employed in the reasoning of some general optimisation algorithms.

By definition, a set K in $\mathbb{R}^n$ is convex if for each pair of points $\mathbf{x}_0, \mathbf{x}_1 \in K$ and for each $\theta \in [0,1]$ also $\mathbf{x}_\theta \in K$, where

$$\mathbf{x}_\theta = (1-\theta)\mathbf{x}_0 + \theta\, \mathbf{x}_1 . \tag{3.160}$$

An equivalent definition is that for any set of points $\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_m \in K \quad \mathbf{x}_\theta \in K$ where

$$\mathbf{x}_\theta = \sum_{i=0}^{m} \theta_i \mathbf{x}_i \ \ and \ \ \sum_{i=0}^{m} \theta_i = 1 \ \ \wedge \ \ \theta_i \geq 0 \ \forall i . \tag{3.161}$$

A convex function on a convex set K is a function for which the epigraph is a convex set. The epigraph of a function is the set of points in $\mathbb{R} \times \mathbb{R}^n$ that lies on or above the graph of the function. The equivalent definition of a convex function $f(\mathbf{x})$ is that for any $\mathbf{x}_0, \mathbf{x}_1 \in K$ it follows that

$$f_\theta \leq (1-\theta) f_0 + \theta\, f_1 \quad \forall \theta \in [0,1]. \tag{3.162}$$

The definition of a strictly convex function is similar but with strict inequality in the above equation. If $-f(\mathbf{x})$ is convex then $f(\mathbf{x})$ is said to be concave.

If f is convex and $\mathbb{C}^1$ on an open convex set *K*, then[1] for each pair $\mathbf{x}_0, \mathbf{x}_1 \in K$

$$f_1 \geq f_0 + \left(\mathbf{x}_1 - \mathbf{x}_0\right)^T \nabla f_0 . \tag{3.163}$$

This means that a graph of *f* must lie above or along its linearisation about any point. It immediately follows (by interchanging $\mathbf{x}_0$ and $\mathbf{x}_1$) that

$$\left(\mathbf{x}_1 - \mathbf{x}_0\right)^T \nabla f_1 \geq f_1 - f_0 \geq \left(\mathbf{x}_1 - \mathbf{x}_0\right)^T \nabla f_0 . \tag{3.164}$$

This corresponds to a statement that the slope of a convex function *f* is non-decreasing along any line. If f is $\mathbb{C}^2$, this result implies (by taking the limit $\left\|\mathbf{x}_1 - \mathbf{x}_0\right\| \to 0$) that $\nabla^2 f\left(\mathbf{x}\right)$ is positive semi-definite at each $\mathbf{x} \in K$.

A convex programming problem is a problem of minimisation of a convex function on a convex set. Such a problem is

$$\begin{array}{ll} \textit{minimise} & f\left(\mathbf{x}\right) \\ \textit{subject to} & \mathbf{x} \in K,\; K = \left\{\, \mathbf{x};\, c_i\left(\mathbf{x}\right) \geq 0,\; i = 1,2, ..., m \,\right\}, \end{array} \tag{3.165}$$

where $f\left(\mathbf{x}\right)$ is convex on *K* and constraint functions $c_i\left(\mathbf{x}\right)$ that define *K* are concave on $\mathbb{R}^n$. Convexity of *K* defined as above follows from the fact that an epigraph of any concave function is a convex set, and from a known theorem that intersection of convex sets is a convex set.

The following important results hold for convex programming problems[1]:

Theorem 3.11:

Every local solution to a convex programming problem is also a global solution, and the set of global solutions S is convex. If $f\left(\mathbf{x}\right)$ is also strictly convex on K, then the solution is unique.

Theorem 3.12:

In the convex programming problem (3.165), if $f(\mathbf{x})$ and $c_i(\mathbf{x})$ are $\mathbb{C}^1$ on K and if the Kuhn-Tucker conditions (3.146) hold at $\mathbf{x}^*$, then $\mathbf{x}^*$ is a global solution to the problem.

3.8.4 Duality in Nonlinear Programming

The concept of duality provides a set of rules for transformation of one problem to another. By applying these rules alternative formulation of the problem is obtained, which is sometimes more convenient computationally or has some theoretical significance. The original problem is referred to as the primal and the transformed problem as the dual. Some duality transformations have a symmetry property that the dual of the dual is the primal (i.e. that the transformation applied twice gives the original problem).

Usually some of the variables in the dual correspond to Lagrange multipliers of the primal and take the value λ^* at the dual solution. The dual and the primal should be related in the way that the dual has a solution from which the solution of the primal can be derived. Duality transformations of this kind are associated with the convex programming problem as the primal. A set of such duality transformations can be derived from the Wolfe dual whose statement is given in the theorem below[1].

Theorem 3.13:

If $\mathbf{x}^*$ solves the primal convex programming problem (3.165), if f and c_i are $\mathbb{C}^1$ functions, and if the regularity assumption (3.145) holds, then $\mathbf{x}^*, \lambda^*$ solve the dual problem

$$\begin{aligned} &\underset{\mathbf{x},\lambda}{\textit{maximise}} && \mathcal{L}(\mathbf{x},\lambda) \\ &\textit{subject to} && \nabla_x \mathcal{L}(\mathbf{x},\lambda) = 0, \quad \lambda \geq 0\,. \end{aligned} \tag{3.166}$$

The minimum primal and maximum dual function values are equal, i.e. $f^* = \mathcal{L}(\mathbf{x}^*, \lambda^*)$

The Wolfe dual is not symmetric. The dual is not necessarily even a convex programming problem. An advantageous property is that if the primal is unbounded then the dual has inconsistent constraints and therefore does not have a solution. It is possible that the primal has inconsistent constraints, but the dual still has a solution. However if the constraints are linear, then infeasible constraints in the primal imply that the dual is unbounded.

An example of application is the quadratic programming problem

$$\begin{aligned} &\textit{minimise} && \tfrac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x}+\mathbf{g}^T\mathbf{x} \\ &\textit{subject to} && \mathbf{A}^T\mathbf{x}\ge\mathbf{b} \end{aligned} \tag{3.167}$$

where $\mathbf{G}$ is positive definite. The Wolfe dual is

$$\begin{aligned} &\underset{\mathbf{x},\lambda}{\textit{maximise}} && \tfrac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x}+\mathbf{g}^T\mathbf{x}-\lambda^T\left(\mathbf{A}^T\mathbf{x}-\mathbf{b}\right) \\ &\textit{subject to} && \mathbf{G}\mathbf{x}+\mathbf{g}-\mathbf{A}\lambda=0,\quad \lambda\ge 0\,. \end{aligned} \tag{3.168}$$

The first set of constraints can be used to eliminate $\mathbf{x}$ (i.e. $\mathbf{x}=\mathbf{G}^{-1}(\mathbf{A}\lambda-\mathbf{g})$), which gives the problem

$$\begin{aligned} &\underset{\lambda}{\textit{maximise}} && -\tfrac{1}{2}\lambda^T\left(\mathbf{A}^T\mathbf{G}^{-1}\mathbf{A}\right)\lambda+\lambda^T\left(\mathbf{b}+\mathbf{A}^T\mathbf{G}^{-1}\mathbf{g}\right)-\tfrac{1}{2}\mathbf{g}^T\mathbf{G}^{-1}\mathbf{g} \\ &\textit{subject to} && \lambda\ge 0\,. \end{aligned} \tag{3.169}$$

This is again a quadratic programming problem, but subject only to simple bounds. When the solution λ^* is found, $\mathbf{x}^*$ is obtained by solving the equation used for elimination of $\mathbf{x}$ from (3.168).

3.9 Quadratic Programming

A quadratic programming (QP) problem is an optimisation problem with quadratic objective function and linear constraint functions, i.e.

$$\textit{minimise} \qquad q(\mathbf{x})=\tfrac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x}+\mathbf{g}^T\mathbf{x}$$

$$\begin{aligned} &\textit{subject to} && \mathbf{a}_i^T\mathbf{x} = b_i,\ i \in E \\ &\textit{and} && \mathbf{a}_i^T\mathbf{x} \ge b_i,\ i \in I\,. \end{aligned} \tag{3.170}$$

where **G** is symmetric. If **G** is positive semi-definite, a local solution $\mathbf{x}^*$ is also global, and if **G** is positive definite, it is also unique. This follows from Theorem 3.13 since such a problem is a convex programming problem. Only if **G** is indefinite can a local solution which is not global occur.

3.9.1 Equality Constraints Problem

The quadratic programming problem with only equality constraints can be stated as

$$\begin{aligned} &\textit{minimise} && q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{g}^T\mathbf{x} \\ &\textit{subject to} && \mathbf{A}^T\mathbf{x} = \mathbf{b}\,. \end{aligned} \tag{3.171}$$

It will be assumed that there are $m \le n$ constraints and that **A** has rank *m*, which ensures that unique multipliers λ^* exist. **A** is a $n \times m$ matrix whose columns are vectors $\mathbf{a}_i$, $i \in E$ from (3.170), and $\mathbf{b} \in \mathbb{R}^m$.

The problem can be transformed to an unconstrained minimisation problem by direct elimination of variables using constraints. Let partitions

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix} \tag{3.172}$$

be defined, where $\mathbf{x}_1 \in \mathbb{R}^m$ and $\mathbf{x}_2 \in \mathbb{R}^{n-m}$, so that $\mathbf{A}_1$ is $m \times m$, $\mathbf{G}_{11}$ is $m \times m$, etc. Then *m* variables in the vector $\mathbf{x}_1$ can be expressed in terms of $\mathbf{x}_2$ as

$$\mathbf{x}_1 = \mathbf{A}_1^{-T}\left(\mathbf{b} - \mathbf{A}_2^T\mathbf{x}_2\right). \tag{3.173}$$

Substituting this into $q(\mathbf{x})$ gives

$$\begin{aligned}&\psi(\mathbf{x}_2) = q(\mathbf{x}_1(\mathbf{x}_2), \mathbf{x}_2) = \\ &\tfrac{1}{2}\mathbf{x}_2^T\left(\mathbf{G}_{22} - \mathbf{G}_{21}\mathbf{A}_1^{-T}\mathbf{A}_2^T - \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{G}_{12} + \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{G}_{11}\mathbf{A}_1^{-T}\mathbf{A}_2^T\right)\mathbf{x}_2 + \\ &\mathbf{x}_2^T\left(\mathbf{G}_{21} - \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{G}_{11}\right)\mathbf{A}_1^{-T}\mathbf{b} + \tfrac{1}{2}\mathbf{b}^T\mathbf{A}_1^{-1}\mathbf{G}_{11}\mathbf{A}_1^{-T}\mathbf{b} + \\ &\mathbf{x}_2^T\left(\mathbf{g}_2 - \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{g}_1\right) + \mathbf{g}_1^T\mathbf{A}_1^{-T}\mathbf{b}\end{aligned} \quad . \tag{3.174}$$

The problem is so transformed to unconstrained minimisation of $\psi(\mathbf{x}_2)$. If the Hessian (the matrix in the round brackets in the second line) is positive definite, then a unique minimiser $\mathbf{x}_2^*$ exists and is obtained by solving the linear system of equations $\nabla\psi(\mathbf{x}_2) = 0$. $\mathbf{x}_1^*$ is obtained by substitution in (3.173). The Lagrange multiplier vector is defined by $\mathbf{g}^* = \mathbf{A}\lambda^*$ where $\mathbf{g}^* = \nabla q(\mathbf{x}^*) = \mathbf{g} + \mathbf{G}\mathbf{x}^*$, and can be calculated by solving the first partition $\mathbf{g}_1^* = \mathbf{A}_1\lambda^*$.

The described approach is not the only possibility. First of all, it is possible to rearrange variables and choose some other set of variables to be independent. More generally a linear transformation of variables can be made. Such a general approach is the generalised elimination method.

Let $\mathbf{Y}$ and $\mathbf{Z}$ be $n \times m$ and $n \times (n-m)$ matrices such that $[\mathbf{Y} : \mathbf{Z}]$ is non-singular and

$$\begin{aligned}\mathbf{A}^T\mathbf{Y} &= \mathbf{I}_{m\times m}, \\ \mathbf{A}^T\mathbf{Z} &= \mathbf{0}_{m\times n-m}.\end{aligned} \tag{3.175}$$

$\mathbf{Y}^T$ can be regarded as the left generalised inverse of $\mathbf{A}$ since a solution of the system $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ is given by $\mathbf{x} = \mathbf{Y}\mathbf{b}$. The solution is not unique and other solutions are given by $\mathbf{x} = \mathbf{Y}\mathbf{b} + \delta$ where δ is in the $n-m$ - dimensional null column space of **A**, i.e.

$$\mathbf{A}^T\delta = 0 \tag{3.176}$$

If the matrix **Z** has linearly independent columns $\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{\mathbf{n-m}}$, then these vectors form a basis of the null space of **A**[29]. At any feasible point **x** (i.e. solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$) any feasible correction δ (which gives another solution) can be written as

$$\delta = \sum_{i=1}^{n-m} y_i\mathbf{z}_i \ , \tag{3.177}$$

where $y_1, y_2, ..., y_{n-m}$ are the components in the null space of **A**, referred to also as reduced variables in this space. Any feasible point can be written as

$$\mathbf{x} = \mathbf{Yb} + \mathbf{Zy} . \tag{3.178}$$

The above equation provides a way of eliminating constraints $\mathbf{Ax} = \mathbf{b}$ by transformation to the $n-m$ - dimensional space of reduced variables in which constraints are always satisfied, and is therefore a generalisation of (3.173). Substituting the equation into (3.171) gives the reduced quadratic function

$$\psi(\mathbf{x}) = \tfrac{1}{2}\mathbf{y}^T\mathbf{Z}^T\mathbf{GZy} + (\mathbf{g} + \mathbf{GYb})^T\mathbf{Zy} + \tfrac{1}{2}(\mathbf{g} + \mathbf{GYb})^T\mathbf{Yb} . \tag{3.179}$$

If the reduced Hessian matrix $\mathbf{Z}^T\mathbf{GZ}$ is positive definite then a unique solution exists and can be obtained by solution of the system $\nabla\psi(\mathbf{y}) = 0$, i.e.

$$(\mathbf{Z}^T\mathbf{GZ})\mathbf{y} = -\mathbf{Z}^T(\mathbf{g} + \mathbf{GYb}). \tag{3.180}$$

It is convenient to solve this system by Choleski factorisation[30],[33], which also enables positive definiteness to be checked. $\mathbf{x}^*$ is then obtained from $\mathbf{y}^*$ by using (3.178). Lagrangian multipliers are obtained from $\mathbf{g}^* = \mathbf{A}\lambda^*$, which after pre-multiplying by $\mathbf{Y}^T$ gives

$$\lambda^* = \mathbf{Y}^T\mathbf{g}^* = \mathbf{Y}^T(\mathbf{Gx}^* + \mathbf{g}). \tag{3.181}$$

Note that $\mathbf{g}$ in this equation does not refer to the gradient vector, but is a constant vector in the definition of $q(\mathbf{x})$. The reduced gradient vector is $\mathbf{Z}^T(\mathbf{g} + \mathbf{GYb})$. This shows that the reduced derivatives can be obtained by pre-multiplication by $\mathbf{Z}^T$, since $\mathbf{g} + \mathbf{GYb} = \nabla q(\mathbf{Yb})$ is the gradient of $q(\mathbf{x})$ at $\mathbf{x} = \mathbf{Yb}$.

Different methods arise from different choice of **Y** and **Z**. It is convenient to use any orthogonal (QR) factorization[26]-[28] of **A**:

$$\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} = [\mathbf{Q}_1\ \mathbf{Q}_2]\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} = \mathbf{Q}_1\mathbf{R} , \tag{3.182}$$

where **Q** is a $n \times n$ orthogonal matrix, **R** is a $m \times m$ upper triangular matrix, and $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are $n \times m$ and $n \times (n-m)$ partitions of **Q**. Then we can choose

$$\mathbf{Y} = \mathbf{Q}_1\mathbf{R}^{-T}, \quad \mathbf{Z} = \mathbf{Q}_2 . \tag{3.183}$$

This implies that the vector $\mathbf{Yb}$ from (3.178) is orthogonal to any feasible change δ and the reduced coordinate directions $\mathbf{z}_i$ are mutually orthogonal.

The reduced system (3.180) is first solved to obtain $\mathbf{y}^*$, and then $\mathbf{x}^*$ is calculated by substitution into (3.178). Numerically it is most convenient to evaluate vector $\mathbf{Yb}$, which appears in these equations, by forward substitution in $\mathbf{R}^T\mathbf{u} = \mathbf{b}$ (since $\mathbf{R}$ is upper triangular) followed by multiplication $\mathbf{Yb} = \mathbf{Q}_1\mathbf{u}$. Multipliers λ^* are then calculated by backward substitution in $\mathbf{R}\lambda^* = \mathbf{Q}_1^T\mathbf{g}^*$. Such a scheme is referred to as the orthogonal factorization method. Its advantage is that because of using orthogonal transformations, the method is less sensitive to round-off errors[27].

In general, $\mathbf{Y}$ and $\mathbf{Z}$ can be obtained by completion of the matrix $\mathbf{A}$ to a full-rank $n \times n$ matrix and partitioning of the inverse of that matrix. For example, we can choose any $n \times (n-m)$ matrix $\mathbf{V}$ such that the matrix $[\mathbf{A} : \mathbf{V}]$ is non-singular. $\mathbf{Y}$ and $\mathbf{Z}$ are then obtained by

$$[\mathbf{A} : \mathbf{V}]^{-1} = \begin{bmatrix} \mathbf{Y}^T \\ \mathbf{Z}^T \end{bmatrix}, \tag{3.184}$$

where $\mathbf{Y}$ and $\mathbf{Z}$ are $n \times m$ and $n \times (n-m)$ partitions respectively. They satisfy conditions (3.175) and are therefore suitable for use in the generalised elimination method. The resulting method can be interpreted as a method which makes linear transformation with the matrix $[\mathbf{A} : \mathbf{V}]$.

Different methods arise from specific choices of $\mathbf{V}$. Choosing

$$\mathbf{V} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{3.185}$$

results in the direct elimination method. The identity

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}_1^{-1} & \mathbf{0} \\ -\mathbf{A}_2\mathbf{A}_1^{-1} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{Y}^T \\ \mathbf{Z}^T \end{bmatrix} \tag{3.186}$$

gives expressions for $\mathbf{Y}$ and $\mathbf{Z}$. It can then be verified by substitution into (3.180) and taking into account the appropriate partitioning that the resulting method is identical to the direct elimination method.

The orthogonal factorization method is obtained by setting

$$\mathbf{V} = \mathbf{Q}_2, \tag{3.187}$$

where $\mathbf{Q}_2$ is defined by (3.182). By using the identity

$$[\mathbf{A} : \mathbf{V}]^{-1} = [\mathbf{Q}_1\mathbf{R} : \mathbf{Q}_2]^{-1} = \begin{bmatrix} \mathbf{R}^{-1}\mathbf{Q}_1^T \\ \mathbf{Q}_2^T \end{bmatrix} = \begin{bmatrix} \mathbf{Y}^T \\ \mathbf{Z}^T \end{bmatrix} \tag{3.188}$$

(3.183) is obtained, which confirms that the orthogonal factorization method was obtained. The above equation can be expressed as

$$[\mathbf{A} : \mathbf{V}]^{-1} = \begin{bmatrix} \mathbf{A}^+ \\ \mathbf{V}^+ \end{bmatrix} \tag{3.189}$$

where $\mathbf{A}^+ = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is the full rank generalised inverse, therefore $\mathbf{Y} = \mathbf{A}^{+T}$ from (3.183).

3.9.2 Active Set Methods

Inequality constraints can not be simply used to eliminate variables or reduce the dimension of the problem. Only those inequality constraints that are active in the solution actually affect matters. If it would be known in advance which constraints are active in the solution, these constraints could be used as equality constraints and all other constraints could be ignored. Active set methods gradually update the set of active constraints and solve the resulting equality constrained problems where constraints regarded as inactive are temporarily ignored. It is assumed that the Hessian matrix of the problem is positive definite. The basic idea is illustrated in Figure 3.12 and described below .

On the k-th iteration a feasible point $\mathbf{x}^{(k)}$ is known which satisfies active constraints as equalities, i.e. $\mathbf{a}_i^T\mathbf{x}^{(k)} = b_i \;\; \forall i \in \mathcal{A}$ where $\mathcal{A}$ is the index set of constraints currently regarded as active and treated as equality constraints. All equality constraints are in this set. $\mathbf{x}^{(k)}$ also satisfies $\mathbf{a}_i^T\mathbf{x}^{(k)} > b_i \;\; \forall i \notin \mathcal{A}$, so that the current active set $\mathcal{A}$ is equivalent to the set of active constraints $\mathcal{A}^{(k)}$.

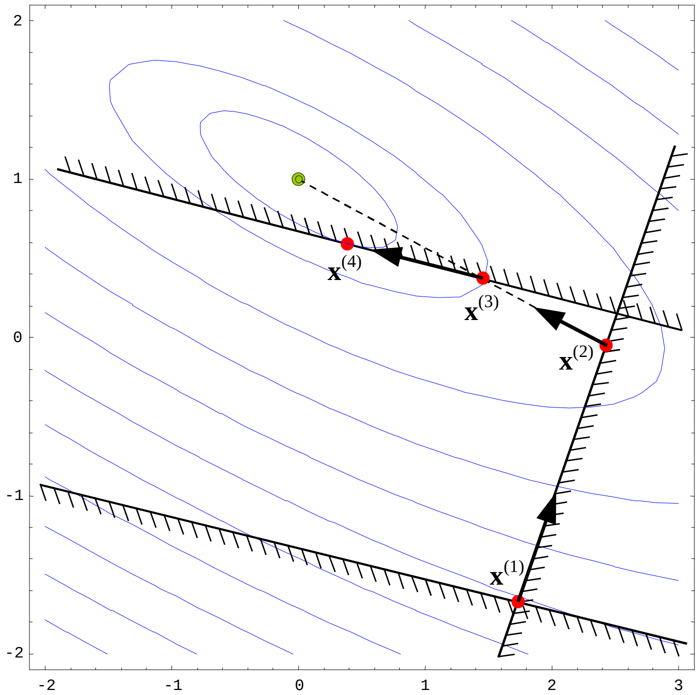

Figure 3.12: Progress of the active set method in a problem with three inequality constraints.

The iteration attempts to solve the equality problem where only active constraints occur. By shifting the origin to $\mathbf{x}^{(k)}$ and looking for a correction $\delta^{(k)}$ this problem is

$$\begin{aligned} &\textit{minimise} && \tfrac{1}{2}\delta^T\mathbf{G}\delta + \delta^T\mathbf{g}^{(k)} \\ &\textit{subject to} && \mathbf{a}^T\delta = 0 \quad \forall\, i \in \mathcal{A}, \end{aligned} \tag{3.190}$$

where $\mathbf{g}^{(k)} = \nabla q\left(\mathbf{x}^{(k)}\right) = \mathbf{g} + \mathbf{G}\mathbf{x}^{(k)}$.

If δ is feasible with respect to constraints not in $\mathcal{A}$, then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta$ is taken. If not a line search is made in the direction $\mathbf{s}^{(k)} = \delta^{(k)}$ to find the best feasible point. $\alpha^{(k)}$ is found which solves

$$\alpha^{(k)} = \min\left(1, \min_{\substack{i \notin \mathcal{A} \wedge \\ \mathbf{a}_i^T \mathbf{s}^{(k)} < 0}} \frac{b_i - \mathbf{a}_i^T \mathbf{x}^{(k)}}{\mathbf{a}_i^T \mathbf{s}^{(k)}}\right) \tag{3.191}$$

and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{s}^{(k)}$ is set.

If $\alpha^{(k)} < 1$ then a new constraint (with index *p*, say) which achieves the minimum in the above equation becomes active and its index *p* is added to the active set $\mathcal{A}$.

If $\mathbf{x}^{(k)}$ solves the current equality problem, then it is possible to compute multipliers $\lambda^{(k)}$ as described in the previous section. Vectors $\mathbf{x}^{(k)}$ and $\lambda^{(k)}$ satisfy all the first order conditions for the original inequality constrained problem except possibly the conditions $\lambda_i \geq 0, i \in I$. The test is therefore made if these conditions are satisfied for all inequality constraints in $\mathcal{A}$. If so, the first order conditions are satisfied and since the problem is convex (because **G** is positive definite), this is sufficient for $\mathbf{x}^{(k)}$ to be a global solution. Otherwise there exists an index *q* such that $\lambda_q^{(k)} < 0$. In this case it is possible to reduce $q(\mathbf{x})$ by allowing constraint *q* to become inactive (according to discussion around equation (3.141)). Constraint *q* is therefore removed from $\mathcal{A}$ and the algorithm continues as before. It is possible that there are more than one indices with $\lambda_j^{(k)} < 0$. Then *q* is selected so that it solves

$$\min_{i \in \mathcal{A} \cap I} \lambda_i^{(k)}. \tag{3.192}$$

The complete algorithm is outlined below.

Algorithm 3.10: The active set method.

A feasible point $\mathbf{x}^{(1)}$ must be given. $\mathcal{A} = \mathcal{A}^{(1)}$ is set where $\mathcal{A}^{(1)}$ contains indices of all constraints for which $c_i\left(\mathbf{x}^{(1)}\right) = 0$. The *k*-th iteration is then as follows:

1. If $\delta = \mathbf{0}$ does not solve (3.190) then go to 3.
2. Compute Lagrange multipliers $\lambda^{(k)}$ and solve (3.192). If $\lambda_q^{(k)} \geq 0$ then terminate with $\mathbf{x}^* = \mathbf{x}^{(k)}$, otherwise remove q from $\mathcal{A}$.
3. Solve (3.190) for $\mathbf{s}^{(k)}$.
4. Solve (3.191) to find $\alpha^{(k)}$ and set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{s}^{(k)}$.
5. If $\alpha^{(k)} < 1$, add p to $\mathcal{A}$.
6. Set $k = k + 1$ and go to 1.

The initial feasible point can be obtained from any given point $\mathbf{x}^{(0)}$ by iteratively solving the problem

$$\begin{aligned} &\textit{minimise} && \sum_{i \in V^{(k)}} \left(b_i - \mathbf{a}_i^T \mathbf{x} \right) \\ &\textit{subject to} && \mathbf{a}_i^T \mathbf{x} \geq b_i \quad \forall\, i \notin V^{(k)}, \end{aligned} \tag{3.193}$$

where $V^{(k)}$ is the set of infeasible constraints at $\mathbf{x}^{(k)}$. Iteration is repeated until $\mathbf{x}^{(k)}$ becomes a feasible point. Minimisations are performed as line searches along edges $\mathbf{s}^{(k)} = \mathbf{a}_q^{(k)}$ where $q^{(k)}$ is the index with the least Lagrange multiplier in iteration k. Each search terminates with a new constraint becoming active[1].

So far it was assumed that the Hessian matrix $\mathbf{G}$ is positive definite. If $\mathbf{G}$ is indefinite then local solutions exist which are not global. For any local solution the reduced Hessian matrix $\mathbf{Z}^T\mathbf{G}\mathbf{Z}$ is positive semi-definite and this matrix is actually used when the equality problem is solved. However, when the algorithm proceeds, not necessarily all constraints that are active in the solution are in the active set. Therefore problem (3.190) with indefinite reduced Hessian can arise. In this case a solution of (3.190) $\delta^{(k)}$ is no longer a minimiser. Any feasible descent direction can be chosen for $\mathbf{s}^{(k)}$, for example the negative reduced gradient vector. $\alpha^{(k)}$ is then obtained from

$$\alpha^{(k)} = \min_{\substack{i \notin \mathcal{A}, \\ \mathbf{a}_i^T \mathbf{s}^{(k)} < 0}} \frac{b_i - \mathbf{a}_i^T \mathbf{x}^{(k)}}{\mathbf{a}_i^T \mathbf{s}^{(k)}} \tag{3.194}$$

rather than from (3.191). If the above equation does not have a solution (i.e. the infimum of the right-hand side is $-\infty$), this indicates that the original QP problem is unbounded.

3.10 Penalty Methods

Penalty methods[6] are a traditional and commonly used approach to constrained minimisation. The idea of penalty methods is to control constraint violations by penalizing them. The original objective function is modified by addition of penalty terms, which monotonically increase as constraint violations increase. The sum of the objective function and penalty terms is called the penalty function. Some parameter is usually associated with penalty terms to control the amount of the penalty. The minimiser of the objective function is approximated by unconstrained minimisers of the penalty function, which should converge to the constrained minimiser as the control parameter is increased (Figure 3.13).

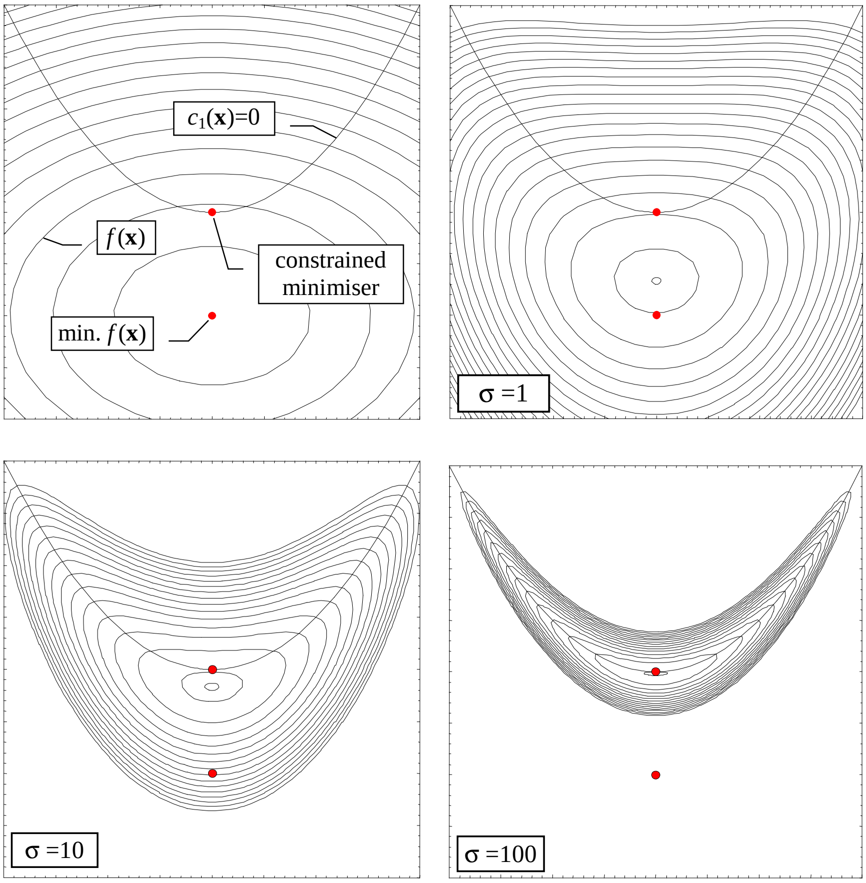

Figure 3.13: Use of penalty functions. The problem with one equality constraint is sketched in the first picture. The subsequent pictures show penalty function contours with increasing parameter σ. The minimiser of the penalty functions approaches the problem solution as σ increases, but also ill-conditioning increases.

The following penalty function can be used for equality constraints:

$$\phi(\mathbf{x},\sigma) = f(\mathbf{x}) + \frac{1}{2}\sigma\sum_{i\in E}\left(c_i^{\,2}\right) = f(\mathbf{x}) + \frac{1}{2}\sigma\sum_{i\in E}\mathbf{c}(\mathbf{x})^T\mathbf{c}(\mathbf{x}). \qquad (3.195)$$

Parameter σ determines the amount of the penalty. A simple penalty algorithm is outlined below.

Algorithm 3.11: The penalty algorithm.

1. Choose a fixed sequence $\left\{\sigma^{(k)}\right\}\to\infty$, e.g. $\{1,10,100,1000,...\}$.
2. Find a local minimiser $\mathbf{x}\left(\sigma^{(k)}\right)$ of $\phi\left(\mathbf{x},\sigma^{(k)}\right)$, using a minimiser of the previous iteration as a starting guess.
3. Terminate if $\mathbf{c}\left(\mathbf{x}\left(\sigma^{(k)}\right)\right)$ is sufficiently small, otherwise go to 2.

The quantities associated with $\sigma^{(k)}$ will be denoted by upper index *k*, e.g. $\mathbf{x}\left(\sigma^{(k)}\right)=\mathbf{x}^{(k)}$, $f\left(\mathbf{x}\left(\sigma^{(k)}\right)\right)=f^{(k)}$, etc. The following convergence result holds for such an algorithm[6]:

Theorem 3.14 (penalty function convergence):

Let $f(\mathbf{x})$ be bounded below on a non-empty feasible region and let global minimisers be evaluated in step 2 of the above algorithm. If $\sigma^{(k)}\to\infty$ monotonically, then $\left\{\phi^{(k)}\left(\mathbf{x}^{(k)},\sigma^{(k)}\right)\right\}$, $\left\{\mathbf{c}^{(k)T}\mathbf{c}^{(k)}\right\}$ and $\left\{f^{(k)}\right\}$ are non-decreasing, $\mathbf{c}^{(k)}\to 0$ and any accumulation point $\mathbf{x}^*$ of $\left\{\mathbf{x}^{(k)}\right\}$ solves the equality constrained problem.

The algorithm has some other limiting properties, which enable useful estimations to be made and are gathered in the theorem below.

Theorem 3.15 (penalty function convergence):

If $\sigma^{(k)}\to\infty$, $\mathbf{x}^{(k)}\to\mathbf{x}^*$ and $rank\,\mathbf{A}^*=m$ (m is the number of constraints), then $\mathbf{x}^*$ is a KT point and the following hold:

$$\lambda^{(k)}=-\sigma^{(k)}\mathbf{c}^{(k)}=\lambda^*+o(1), \tag{3.196}$$

$$f^*=\phi^{(*)}=\phi^{(k)}+\tfrac{1}{2}\sigma^{(k)}\mathbf{c}^{(k)T}\mathbf{c}^{(k)}+o(1/\sigma), \tag{3.197}$$

$$\mathbf{h}^{(k)}=\frac{-\mathbf{T}^*\lambda^*}{\sigma^{(k)}}+o(1/\sigma), \tag{3.198}$$

where $\mathbf{T}$ is defined by

$$\begin{bmatrix} \mathbf{W}^* & -\mathbf{A}^* \\ -\mathbf{A}^{*T} & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{H}^* & -\mathbf{T}^* \\ -\mathbf{T}^{*T} & \mathbf{U}^* \end{bmatrix}, \tag{3.199}$$

$\mathbf{W}$ is the Hessian matrix of the Lagrangian function and $\mathbf{A}$ is the Jacobian matrix of constraints. Notation $a = o(h) \Leftrightarrow a/h \to 0$ has been used.

These results enable some enhancements of the algorithm to be made. (3.196) gives an estimation of the Lagrange multipliers and (3.197) can be used as a better estimation to f^* than $\phi^{(k)}$ itself. (3.199) can be used to terminate the iteration and to provide better initial approximations when minimising $\phi\left(\mathbf{x}, \sigma^{(k)}\right)$.

For inequality constraint problems the following penalty function can be used:

$$\phi(\mathbf{x}, \sigma) = f(\mathbf{x}) + \frac{1}{2}\sigma \sum_{i \in I} \left(\min\left(0, c_i(\mathbf{x})\right)\right)^2 . \tag{3.200}$$

A disadvantage of this penalty function is the jump discontinuity in second derivatives where $c_i(\mathbf{x}) = 0$. $\mathbf{x}^{(k)}$ approaches $\mathbf{x}^*$ from the infeasible side of the inequality constraints, therefore algorithms that use such a penalty function are called exterior point algorithms.

Another class of algorithms for inequality constraints are barrier function methods. Barrier terms, which are infinite on the constraint boundaries are added to the penalty function. These algorithms preserve strict constraint feasibility in all iterations and are therefore called interior point algorithms. Their use is advantageous when the objective function is not defined in the infeasible region.

Two commonly used barrier functions are the inverse barrier function

$$\phi(\mathbf{x}, r) = f(\mathbf{x}) + r \sum_{i \in I} \frac{1}{c_i(\mathbf{x})} \tag{3.201}$$

and the logarithmic barrier function

$$\phi(\mathbf{x}, r) = f(\mathbf{x}) - r \sum_{i \in I} \ln\left(c_i(\mathbf{x})\right). \tag{3.202}$$

A sequence $\{r^{(k)}\} \to 0$ is chosen, which ensures that the barrier terms become more and more negligible as compared to the objective function, except close to the constraint boundary. Sequential minimisation of the penalty functions is performed as in Algorithm 3.11.

Penalty and barrier approaches have a simple extension for problems with mixed equality and inequality constraints. Mixed penalty terms for equality constraints and penalty or barrier terms for inequality constraints are added to the objective function for corresponding constraints[6],[14].

The described algorithms are linearly convergent. A difficulty associated with the penalty and barrier approach is that when the control parameter σ is increased (or r decreased, respectively), the Hessian of the penalty (or barrier) function becomes increasingly ill-posed, which is evidently illustrated in Figure 3.13. It is therefore difficult to find accurate solutions of the subsequent unconstrained minimisation problems. The additional problem with barrier functions is that they are not defined in the infeasible region, which can be difficult to handle for minimisation algorithms.

3.10.1 Multiplier Penalty Functions

The multiplier penalty functions represent an attempt to use penalty functions adequately by keeping the control parameter σ finite and thus to avoid ill-conditioning of the penalty function when σ is large. The approach follows from the idea that the constrained minimiser $\mathbf{x}^*$ can be made an unconstrained minimiser of $\phi(\mathbf{x},\sigma)$ by changing the origin of the penalty terms. This leads to the penalty function

$$\begin{gathered}\phi(\mathbf{x},\theta,\sigma) = f(\mathbf{x}) + \tfrac{1}{2}\sum_{i \in E} \sigma_i \left(c_i(\mathbf{x}) - \theta_i\right)^2 = \\ f(\mathbf{x}) + \tfrac{1}{2}\left(\mathbf{c}(\mathbf{x}) - \theta\right)^T \mathbf{S}\left(\mathbf{c}(\mathbf{x}) - \theta\right)\end{gathered}, \tag{3.203}$$

where $\theta, \sigma \in \mathbb{R}^n$ and $\mathbf{S} = diag(\sigma_i)$ is a diagonal matrix with $S_{ii} = \sigma_i$, and the equality constrained problem is considered. The aim of the algorithm is to find the optimal shift of the origin θ such that a minimiser of $\phi(\mathbf{x},\theta,\sigma)$ with respect to variables $\mathbf{x}$ will correspond to the constrained minimiser $\mathbf{x}^*$.

Let us introduce different parameters

$$\lambda_i = \theta_i \, \sigma_i, \quad i = 1, 2, ..., m. \tag{3.204}$$

If we ignore the term $\frac{1}{2}\sum \sigma_i \theta_i^{\,2}$, which is independent of $\mathbf{x}$ and therefore does not affect the minimiser, ϕ becomes

$$\phi(\mathbf{x}, \lambda, \sigma) = f(\mathbf{x}) - \lambda^T \mathbf{c}(\mathbf{x}) + \tfrac{1}{2} \mathbf{c}(\mathbf{x})^T \mathbf{S}\, \mathbf{c}(\mathbf{x}). \tag{3.205}$$

Because the above function is obtained from (3.195) by adding a multiplier term $-\lambda^T \mathbf{c}$, it is referred to as the multiplier penalty function[1]. There exists optimum values of multipliers λ, for which $\mathbf{x}^*$ minimises $\phi(\mathbf{x}, \lambda, \sigma)$. It turns out that these values are the Lagrange multipliers λ^* at the solution, provided that parameters σ_i are large enough. An exact formulation of this is given in the theorem below[1].

Theorem 3.16:

If second order sufficient conditions for a constrained local minimum hold at $\mathbf{x}^*$, λ^*, then there exists $\sigma' \geq \mathbf{0}$ (i.e. $\sigma_i' \geq 0 \; \forall i$) such that for any $\sigma > \sigma'$ $\mathbf{x}^*$ is an isolated local minimiser of $\phi(\mathbf{x}, \lambda^*, \sigma)$, i.e. $\mathbf{x}^* = \mathbf{x}(\lambda^*)$.

Illustration of the multiplier penalty function is shown in Figure 3.14. This is done for the same problem as in Figure 3.13, so that the multiplier penalty function can be compared to the standard penalty function. The optimal value λ^* was used in the figure and both values of σ_1 were sufficiently large, so that the minimum of the penalty function corresponds to the solution of the original equality constrained problem.

The Lagrange multipliers at the solution of the original problem are not known in advance, therefore a method for generating a sequence $\lambda^{(k)} \to \lambda^*$ must be incorporated in the algorithm.

[1] The term augmented Lagrangian function is also used, since the function can be considered as the Lagrangian function where *f* is augmented by the term $\frac{1}{2}\mathbf{c}(\mathbf{x})^T \mathbf{S}\, \mathbf{c}(\mathbf{x})$.

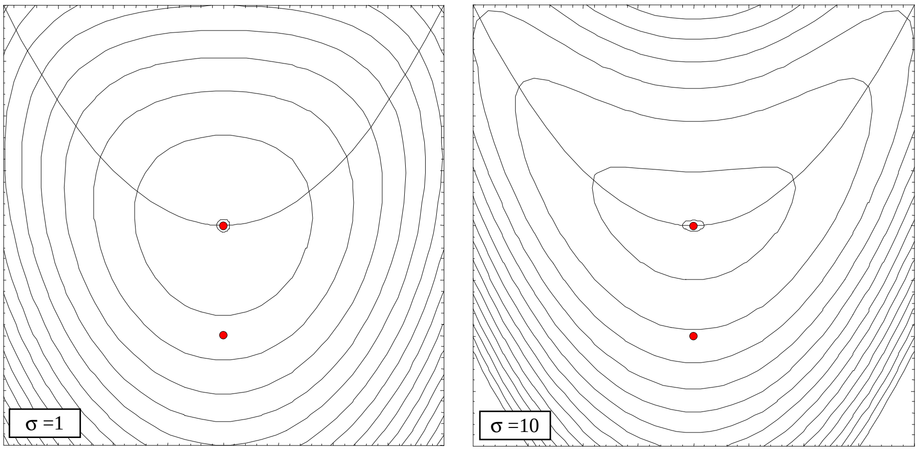

Figure 3.14: Multiplier penalty functions for problem illustrated in Figure 3.13. Minimiser of the multiplier penalty functions corresponds to the constrained minimiser even for a smaller value $\sigma_1 = 1$.

To construct such a method it is assumed that the second order sufficient conditions hold at $\mathbf{x}^*$ and that components of vector σ are sufficiently large. Consider $\mathbf{x}$ being implicitly dependent on λ in a way that $\mathbf{x}(\lambda)$ is a minimiser of $\phi(\mathbf{x}, \lambda)$. Since $\mathbf{x}^* = \mathbf{x}(\lambda^*)$ is by Theorem 3.16 an isolated local minimiser of $\phi(\mathbf{x}, \lambda^*)$, $\mathbf{x}(\lambda)$ is defined uniquely in some neighbourhood Ω_λ of λ^*. $\mathbf{x}(\lambda)$ can be determined by solving the equations

$$\nabla\phi(\mathbf{x}, \lambda) = 0. \tag{3.206}$$

Consider the function

$$\psi(\lambda) = \phi(\mathbf{x}(\lambda), \lambda). \tag{3.207}$$

Since $\mathbf{x}(\lambda)$ is a local minimum of $\phi(\mathbf{x}, \lambda)$, it follows that

$$\psi(\lambda) = \phi(\mathbf{x}(\lambda), \lambda) \le \phi(\mathbf{x}^*, \lambda) = \phi(\mathbf{x}^*, \lambda^*) = \psi(\lambda^*), \tag{3.208}$$

where $\phi(\mathbf{x}^*, \lambda) = \phi(\mathbf{x}^*, \lambda^*)$ is obtained by using $\mathbf{c}^* = 0$ (feasibility) in (3.205). We have $\psi(\lambda) \le \psi(\lambda^*)$, therefore λ^* is a local unconstrained maximiser of $\psi(\lambda)$, and this is true globally if $\mathbf{x}(\lambda)$ is a global minimiser of $\phi(\mathbf{x}, \lambda)$. A sequence $\lambda^{(k)} \to \lambda^*$

can be generated by applying an unconstrained minimisation method to $-\psi(\lambda)$, for which derivatives of ψ with respect to λ are needed.

Derivatives of ϕ with respect to $\mathbf{x}$ are obtained from (3.205):

$$\nabla\phi(\mathbf{x},\lambda,\sigma)=\mathbf{g}-\mathbf{A}\,\lambda+\mathbf{ASc} \tag{3.209}$$

and

$$\mathbf{W}_\sigma=\nabla^2\phi(\mathbf{x},\lambda,\sigma)=\nabla^2 f-\sum_{i\in E}(\lambda_i-\sigma_i c_i)\nabla^2 c_i+\mathbf{ASA}^T\,. \tag{3.210}$$

By the chain rule we have

$$\frac{d\psi}{d\lambda}=\frac{\partial\psi}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\lambda}+\frac{\partial\phi}{\partial\lambda},$$

and since $\partial\phi/\partial\mathbf{x}=\mathbf{0}$ from (3.206) and $\partial\phi/\partial\lambda_i=-c_i$ from (3.205), it follows that

$$\nabla_\lambda\psi\lambda=-\mathbf{c}(\mathbf{x}(\lambda)). \tag{3.211}$$

By the chain rule we then have

$$\frac{d\mathbf{c}}{d\lambda}=\frac{\partial\mathbf{c}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\lambda}=\mathbf{A}^T\,\frac{\partial\mathbf{x}}{\partial\lambda}\,.$$

Applying $d/d\lambda$ to (3.206) gives

$$\frac{d(\nabla\phi)}{d\lambda}=\frac{\partial(\nabla\phi)}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\lambda}+\frac{\partial(\nabla\phi)}{\partial\lambda}=\mathbf{0}\,.$$

$\partial(\nabla\phi)/\partial\mathbf{x}=\nabla^2\phi=\mathbf{W}_\sigma$ and $\partial(\nabla\phi)/\partial\lambda=-\mathbf{A}$ from (3.209), therefore $\dfrac{\partial\mathbf{x}}{\partial\lambda}=\mathbf{W}_\sigma^{-1}\mathbf{A}$ and

$$\nabla_\lambda^2\psi(\lambda)=-\frac{d\mathbf{c}}{d\lambda}=-\,\mathbf{A}^T\mathbf{W}_\sigma^{-1}\mathbf{A}\Big|_{\mathbf{x}(\lambda)}\,. \tag{3.212}$$

The sequence $\lambda^{(k)}\rightarrow\lambda^*$ can be obtained by applying Newton's method from some initial estimate $\lambda^{(1)}$, which gives

$$\lambda^{(k+1)} = \lambda^{(k)} - \left(\left(\mathbf{A}^T \mathbf{W}_\sigma^{-1} \mathbf{A}\right)^{-1} \mathbf{c}\right)\Big|_{\mathbf{x}\left(\lambda^{(k)}\right)}, \tag{3.213}$$

which requires second derivatives of f and **c**. When only first derivatives are available, a quasi-Newton method can be used to find $\mathbf{x}\left(\lambda^{(k)}\right)$ and the resulting **H** matrix can be used to approximate $\mathbf{W}_\sigma^{-1}$ in the above equation, i.e.

$$\lambda^{(k+1)} = \lambda^{(k)} - \left(\left(\mathbf{A}^T \mathbf{H} \mathbf{A}\right)^{-1} \mathbf{c}\right)\Big|_{\mathbf{x}\left(\lambda^{(k)}\right)}, \tag{3.214}$$

An algorithm that uses the derived results is described below.

Algorithm 3.12: The Multiplier penalty algorithm.

1. Set $\lambda = \lambda^{(1)}$, $\sigma = \sigma^{(1)}$, $k = 0$ and $\left\|\mathbf{c}^{(0)}\right\|_\infty = \infty$.
2. Find the minimiser $\mathbf{x}(\lambda,\sigma)$ of $\phi(\mathbf{x},\lambda,\sigma)$ and evaluate $\mathbf{c} = \mathbf{c}(\mathbf{x}(\lambda,\sigma))$.
3. If $\|\mathbf{c}\|_\infty > \frac{1}{4}\left\|\mathbf{c}^{(k)}\right\|_\infty$ then set $\sigma_i = 10\sigma_i \ \forall i : |c_i| > \frac{1}{4}\|\mathbf{c}\|_\infty$ and go to 2.
4. Set $k = k+1$, $\lambda^{(k)} = \lambda$, $\sigma^{(k)} = \sigma$ and $\mathbf{c}^{(k)} = \mathbf{c}$.
5. Evaluate $\lambda^{(k)}$ according to (3.214) (where **H** and **A** are known from step 2) and go to 2..

The aim of line 3 in the above algorithm is to achieve linear convergence at rate $1/4$ or better. The required rate of convergence is obtained when parameters σ are sufficiently large. σ remains constant then and only the parameters λ are changed.

The multiplier penalty function for the inequality constrained case can be derived and used in a similar way[1],[2].

The use of multiplier penalty methods is a significant improvement as compared with the traditional penalty methods. High accuracy of the constrained minimum can be achieved at low values of penalty parameters σ. Ill-conditioning of the minimised penalty function, which is a serious obstacle when using the traditional penalty approach, is avoided to a great extent. An advantage inherited from the penalty approach is that any type of existing unconstrained minimisation techniques can be directly employed in the algorithm. However, the sequential nature of the penalty approach is less efficient than the more direct approach of the sequential quadratic programming approach described in the next section.

3.11 Sequential Quadratic Programming

The penalty approach to constrained optimisation is based on definition of a sequence of unconstrained problems whose solutions converge to the solution of the original problem. A more direct approach is based on approximations of the objective and constraint functions. This seems to be a more efficient approach and many recent developments in optimisation algorithms are related to this approach[15]-[22].

Consider first the equality constrained problem. A system (3.132) is a stationary point condition for a local solution $\mathbf{x}^*$ and Lagrange multipliers in the solution λ^*. By applying Newton's method to solve this stationary point problem the following iteration is obtained:

$$\left[\overline{\nabla}^2 \mathcal{L}^{(k)}\right]\begin{bmatrix}\delta\mathbf{x}\\ \delta\lambda\end{bmatrix} = -\overline{\nabla}\mathcal{L}^{(k)}, \tag{3.215}$$

where $\overline{\nabla}^2\mathcal{L}$ is the matrix of second derivatives of the Lagrangian functions with respect to variables $\mathbf{x},\lambda$ and $\overline{\nabla}\mathcal{L}$ is defined by (3.133). The resulting method is referred to as the Lagrange-Newton method when applied to the solution of an equality constrained problem.

Expressions for the first and second order derivatives are obtained from (3.131). By taking into account these expressions, (3.215) becomes

$$\begin{bmatrix}\mathbf{W}^{(k)} & -\mathbf{A}^{(k)}\\ -\mathbf{A}^{(k)T} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\lambda\end{bmatrix} = \begin{bmatrix}-\mathbf{g}^{(k)} + \mathbf{A}^{(k)}\lambda^{(k)}\\ \mathbf{c}^{(k)}\end{bmatrix} \tag{3.216}$$

$\mathbf{A}^{(k)}$ is the Jacobian matrix of constraints evaluated at $\mathbf{x}^{(k)}$ and $\mathbf{W}^{(k)} = \nabla_x{}^2\left(\mathbf{x}^{(k)},\lambda^{(k)}\right)$ is the Hessian matrix of the Lagrangian function with respect to variables $\mathbf{x}$, i.e.

$$\mathbf{W}^{(k)} = \nabla^2 f\left(\mathbf{x}^{(k)}\right) - \sum_{i\in E}\lambda_i^{(k)}\nabla^2 c_i\left(\mathbf{x}^{(k)}\right). \tag{3.217}$$

The system (3.216) can be rearranged to be solved for $\lambda^{(k+1)} = \lambda^{(k)} + \delta\lambda$ instead of $\delta\lambda$. If we write $\delta^{(k)} = \delta\mathbf{x}$, the system becomes

$$\begin{bmatrix} \mathbf{W}^{(k)} & -\mathbf{A}^{(k)} \\ -\mathbf{A}^{(k)T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta \\ \lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{g}^{(k)} \\ \mathbf{c}^{(k)} \end{bmatrix}. \tag{3.218}$$

Solution of the system gives $\lambda^{(k+1)}$ and $\delta^{(k)}$, while $\mathbf{x}^{(k+1)}$ is obtained by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k)}. \tag{3.219}$$

Similarly as in the case of the Newton Method for unconstrained problems, the system of equations in the Lagrange-Newton iteration can be considered as a solution of a minimisation problem. Consider the problem

$$\begin{aligned} &\underset{\delta}{\textit{minimize}} && q^{(k)}(\delta) = \frac{1}{2}\delta^T \mathbf{W}^{(k)} \delta + \mathbf{g}^{(k)T}\delta + f^{(k)} \\ &\textit{subject to} && \mathbf{l}^{(k)}(\delta) = \mathbf{A}^{(k)T}\delta + \mathbf{c}^{(k)} = 0\,. \end{aligned} \tag{3.220}$$

This can be considered as an approximation of the original problem where the objective function is approximated by the second order Taylor approximation with the addition of constraint curvature terms in the Hessian, and constraints are approximated by the first order Taylor approximation about $\mathbf{x}^{(k)}$. The problem can be solved sequentially, which results in the sequential quadratic programming method summarised below:

Algorithm 3.13: Sequential quadratic programming.

For $k = 1, 2, \ldots$

1. Solve (3.220) for $\delta^{(k)}$. Set $\lambda^{(k+1)}$ to the vector of Lagrange multipliers of the linear constraints.
2. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta$.

The first order conditions $\overline{\nabla}\mathcal{L} = 0$ for this problem give (3.218), therefore the solution $\delta^{(k)}$ of the system (3.218) is a stationary point of the Lagrangian function of the problem (3.220). Following the discussion in section 3.9.1, the second order sufficient conditions for (3.220) are that the reduced Hessian matrix $\mathrm{Z}^{(k)T}\mathbf{W}^{(k)}\mathbf{Z}^{(k)}$ is positive definite. If this is true, then $\delta^{(k)}$ minimises (3.220). It follows that if unique minimisers exist in Algorithm 3.13 for each *k*, then the iteration sequence is identical to that given by the Lagrange-Newton method (3.218) and (3.219). The Lagrange-Newton method can converge to a KT point which is not a minimiser, therefore the sequential quadratic programming algorithm is preferred.

The algorithm can be generalised for solving inequality constrained problems. In this case the subproblem

$$\begin{aligned} &\underset{\delta}{\textit{minimize}} && q^{(k)}(\delta) = \frac{1}{2}\delta^T \mathbf{W}^{(k)}\delta + \mathbf{g}^{(k)T}\delta + f^{(k)} \\ &\textit{subject to} && \mathbf{l}^{(k)}(\delta) = \mathbf{A}^{(k)T}\delta + \mathbf{c}^{(k)} \geq 0\,. \end{aligned} \tag{3.221}$$

is solved instead of (3.220).

The Lagrange-Newton and SQP algorithms have good local convergence properties stated in the following theorem[1]:

Theorem 3.17:

If $\mathbf{x}^{(1)}$ is sufficiently close to $\mathbf{x}^*$, if the Lagrangian matrix

$$\nabla^2 \mathcal{L}^{(1)} = \begin{bmatrix} \mathbf{W}^{(1)} & -\mathbf{A}^{(1)} \\ -\mathbf{A}^{(1)T} & \mathbf{0} \end{bmatrix}$$

is non-singular and if second order sufficient conditions hold at $\mathbf{x}^*, \lambda^*$ with $\textit{rank}\ \mathbf{A}^* = m$ (where m is the number of constraints), then the Lagrange-Newton iteration converges with second order. If $\lambda^{(1)}$ is such that (3.220) is solved uniquely by $\delta^{(1)}$, then the same is true for the SQP method.

The Hessian matrix of the Lagrangian function $\mathbf{W}$ is required in the SQP method. It is possible to approximate $\mathbf{W}$ by using updating formulae[1],[18], analogous to those in quasi-Newton methods. For example, a matrix $\mathbf{B}^{(k)}$ that approximates $\mathbf{W}^{(k)}$ can be updated according to the DFP or BFGS formula, but with

$$\gamma^{(k)} = \nabla\mathcal{L}\left(\mathbf{x}^{(k)}, \lambda^{(k+1)}\right) - \nabla\mathcal{L}\left(\mathbf{x}^{(k)}, \lambda^{(k+1)}\right). \tag{3.222}$$

The resulting algorithms are superlinearly convergent.

The main difficulty of the SQP algorithm as stated above is lack of global convergence properties. The algorithm can fail to converge remote from the solution and it is possible that in some iteration the solution of the subproblem (3.220) or (3.221) does not even exist. The reason for this is essentially the same as for any method which constructs estimates purely on the basis of some simplified models (as for example Newton's method for unconstrained minimisation), i.e. the model is in general adequate only in a limited region which does not necessarily contain the problem solution.

While the line search strategy is a common approach to ensure global convergence of the unconstrained minimisation algorithms, this approach is less applicable in the direct methods for constrained minimisation (except those which solve a sequence of unconstrained subproblems). The reason for this is that especially when non-linear equality constraints are present, any straight line from the current iterate will typically have only one feasible point, which makes use of the line search in a standard way impossible.

The other approach for inducing global convergence is the trust region approach. By adding a step length restriction $\|\delta\| \le h^{(k)}$ to (3.220) or (3.221) the possibility of an unbounded correction is removed. The difficulty is that if $\mathbf{x}^{(k)}$ is infeasible and $h^{(k)}$ is sufficiently small, then the resulting subproblem may not have any feasible points. Another way to ensure that the resulting subproblem is not unbounded is to add the Levenberg-Marquardt term $\nu \mathbf{I}$ to $\mathbf{W}^{(k)}$. It is possible to make $\mathbf{W}^{(k)}$ positive definite by sufficiently increasing the parameter ν [4].

A way of avoiding the difficulties with step length restriction is use of the L_1 exact penalty function[1],[4] in conjunction with the SQP method. An exact penalty function is a penalty function whose unconstrained local minima correspond to constrained local minima of the original problem. The L_1 exact penalty function for a general constrained problem is given by

$$\phi(\mathbf{x}) = \nu\, f(\mathbf{x}) + \sum_{i \in E} |c_i(\mathbf{x})| + \sum_{i \in I} \max(0, -c_i(\mathbf{x})). \tag{3.223}$$

Where ν is a control parameter that weights the relative contribution of $f(\mathbf{x})$ and the penalty terms. If $\mathbf{a}_i^*$ are linearly independent, if $\nu < 1/\|\lambda_i\|_\infty$ and if $\mathbf{x}^*$ satisfies the second order sufficient conditions for the original problem, then $\mathbf{x}^*$ is a local minimiser of (3.223) and can be obtained by a single unconstrained minimisation. The disadvantage of such a penalty function is that it has discontinuous first derivatives on the border of the feasible region (Figure 3.15), which requires the use of special techniques for non-smooth minimisation.

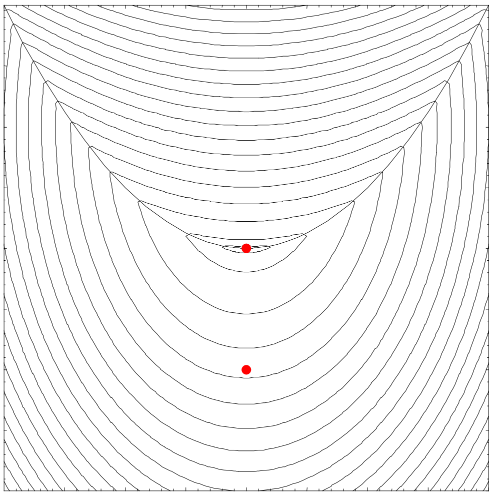

Figure 3.15: L_i exact penalty functions for problem illustrated in Figure 3.13.

For use with the SQP method, approximations (3.220) and (3.221) are substituted in (3.223) and the step restriction (in L_∞ norm) is added, which yields the subproblem

$$\begin{aligned} &\underset{\delta}{\mathit{minimize}} \quad \psi^{(k)}(\delta) = \nu\, q^{(k)}(\delta) + \sum_{i\in E}\left|l_i^{(k)}(\delta)\right| + \sum_{i\in I}\max\left(0, -l_i^{(k)}(\delta)\right) \\ &\mathit{subject\ to} \qquad \|\delta\|_\infty \le h^{(k)} \end{aligned} \tag{3.224}$$

This is an example of a so called L_1QP problem, for which effective algorithms exist[1]. Algorithm 3.13 that solves the subproblem (3.224) is consequently referred to as the SL_1QP algorithm.

The difficulties with an infeasible subproblem when using the step restriction are avoided by using the exact penalty function. The radius of the trust region $h^{(k)}$ is adjusted adaptively in a similar way as in restricted step algorithms for unconstrained minimisation.

Most of the difficulties related to use of the L_1QP subproblem arise from lack of smoothness. The derivative discontinuities give rise to grooves in the penalty surface, which can be difficult to follow by an algorithm. Another problem related to derivative discontinuities is the Maratos effect[1],[21], in which although $\mathbf{x}^{(k)}, \lambda^{(k)}$ may be arbitrarily close to the solution, the SL_1QP method fails to reduce the L_1 exact penalty function. To avoid the effect, the step $\delta^{(k)}$ must be recalculated after making the correction for the higher order errors that arise.

The SQP method and its variants seem to be among the most promising methods for solving general nonlinear problems. A variant of the method developed by A. Tits, E.R. Panier, J. Zhou and C. Lawrence[22],[23] is built in the optimisation shell described in the next chapter.

3.12 Further Remarks

In the present chapter some of the basis of nonlinear programming is outlined. This knowledge is important for understanding the practical requirements for implementation of the algorithmic part in the optimisation shell. The literature cited in this chapter is mostly related to the mathematical and algorithmic background of optimisation and less to practical implementation (except references [3], [8] and [26]). Some implementation aspects are stressed in the next chapter within a larger framework of the optimisation shell. The need for hierarchical and modular implementation, which is stated there, is partially based on the heterogeneity of optimisation algorithms evident from the present chapter.

In practice it is not always obvious which algorithm to use in a given situation. This depends first of all on the case being solved. Although the theory can offer substantial support for making the judgment, most of the literature on optimisation methods recognize the significance of numerical experimentation alongside the theoretical development. This implies a significant aspect that was borne in mind during development of the optimisation shell. The shell should not only include a certain number of algorithms, but also provide an open framework for incorporation of new algorithms and testing them on simple model functions as well as on practical problems.

Many issues important for engineering practice were not taken into account. One of them is handling multiple conflicting optimisation criteria, i.e. solving the problem stated as

$$\begin{aligned} &\textit{minimise} && \left[f_1(\mathbf{x}),\, f_2(\mathbf{x}), \ldots,\, f_m(\mathbf{x}) \right] \\ &\textit{subject to} && \mathbf{x} \in \Omega\,. \end{aligned} \tag{3.225}$$

A common approach is to weight the individual criteria, which leads to the problem

$$\begin{aligned} &\textit{minimise} && f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_m f_m(\mathbf{x}) \\ &\textit{subject to} && \mathbf{x} \in \Omega\,, \end{aligned} \tag{3.226}$$

where w_1, …, w_m are positive weighting coefficients. The problem which arises is how to choose these coefficients. The choice is made either on the basis of experience or in an iterative process where optimisation is performed several times and coefficients are varied on the basis of the optimisation results.

Sometimes it is more convenient to designate one criterion as a primary objective and to constrain the magnitude of the others, e.g. in the following way:

$$\begin{aligned} &\textit{minimise} && f_1(\mathbf{x}) \\ &\textit{subject to} && f_2(\mathbf{x}) \le C_2\,, \\ & && \ldots \\ & && f_m(\mathbf{x}) \le C_m\,, \\ & && \mathbf{x} \in \Omega\,. \end{aligned} \tag{3.227}$$

This approach suffers for a similar defect as weighting criteria, i.e. the solution depends on the choice of coefficients C_2, …, C_m. Attempts to overcome this problem lead to consideration of Pareto optimality[9],[14] and solution of the min-max problem[9],[20].

Another important practical issue is optimisation in the presence of numerical noise. Most of the methods considered in this chapter are designed on the basis of certain continuity assumptions and do not perform well if the objective and constraint functions contain a considerable amount of noise. This can often not be avoided due to complexity of the applied numerical models and their discrete nature (e.g. adaptive mesh refinement in the finite element simulations).

A promising approach to optimisation in the presence of noise incorporates approximation techniques[35],[36]. In this approach successive low order approximations of the objective and constraint functions are made locally on the basis of sampled function values and/or derivatives. This leads to a sequence of approximate optimisation subproblems. They refer to minimisation of the approximate objective functions subject to the approximate constraints and to additional step restriction, which restricts the solution of the subproblem to the region where the approximate functions are adequate. The subproblems are solved by standard nonlinear programming methods. For approximations more data is usually sampled than the minimum amount necessary for determination of the coefficients of the approximate functions, which levels out the effect of noise. A suitable strategy must be defined for choosing the limits of the search region and for the choice of sampling points used for approximations (i.e. the plan of experiments)[35].

A common feature of all methods mentioned in this chapter is that they at best find a local solution of the optimisation problem. There are also methods which can (with a certain probability) find the global solution or more than one local solution at once. The most commonly used are simulated annealing[26],[9],[14] and genetic algorithms[9],[14]. Most of these methods are based on statistical search, which means that they require a large number of function evaluations in order to accurately locate the solution. This makes them less convenient for use in conjunction with expensive numerical simulations, except in cases where global solutions are highly desirable. Use of these techniques can also be suitable for finding global solutions of certain optimisation problems which arise as sub-problems in optimisation algorithms and in which the objective and constraint functions are not defined implicitly through a numerical simulation.

References:

[1] R. Fletcher, *Practical Methods of Optimization (second edition)*, John Wiley & Sons, New York, 1996).

[2] E. J. Beltrami, *An Algorithmic Approach to Nonlinear Analysis and Optimization*, Academic Press, New York, 1970

[3] J. E. Dennis (Jr.), R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[4] D. P. Bertsekas, *Nonlinear Programming (second edition),* Athena Scientific, Belmont, 1999.

[5] *Mathematical Optimization*, electronic book at http://csep1.phy.ornl.gov/CSEP/MO/MO.html , Computational Science Education Project, 1996.

[6] A. V. Fiacco, G. P. McCormick, *Nonlinear Programming – Sequential Unconstrained Minimisation Techniques,* Society for Industrial and Applied Mathematics, Philadelphia, 1990.

[7] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Athena Scientific, Belmont, 1996.

[8] J. L. Nazareth, *The Newton – Cauchy Framework – A Unified Approach to Unconstrained Nonlinear Minimisation,* Springer – Verlag, Berlin, 1994.

[9] A. D. Belgundu, T. R. Chandrupatla: *Optimization Concepts and Applications in Engineering*, Prentice Hall, New Jersy, 1999.

[10] P. E. Gill, W. Murray, M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[11] M. J. D. Powell (editor), *Nonlinear Optimization – Proceedings of the NATO Advanced Research Institute, Cambridge, July 1981*, Academic Press, London, 1982.

[12] M. H. Wright, *Direct Search Methods: Once Scorned, Now Respectable*, in D. F. Griffiths and G. A. Watson (eds.), Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis, p.p. 191 – 208, Addison Wesley Longman, Harlow, 1996.

[13] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Helderman-Verlag, 1988.

[14] S.R. Singiresu, *Engineering Optimization – Theory and Practice (third edition)*, John Wiley & Sons, New York, 1996.

[15] E. Panier, A. L. Tits, *On Combining Feasibility, Descent and Superlinear Convergence in Inequality Constrained Optimization*, Mathematical Programming, Vol. 59 (1993), p.p. 261 - 276.

[16] C. T. Lawrence, A. L. Tits, *Nonlinear Equality Constraints in Feasible Sequential Quadratic Programming*, Optimization Methods and Software, Vol. 6, 1996, pp. 265 - 282.

[17] J. L. Zhou, A. L. Tits, *An SQP Algorithm for Finely Discretized Continuous Minimax Problems and Other Minimax Problems With Many Objective Functions*, SIAM Journal on Optimization, Vol. 6, No. 2, 1996, pp. 461 - 487.

[18] P. Armand, J. C. Gilbert, *A piecewise Line Search Technique for Maintaining the Positive Definiteness of the Matrices in the SQP Method*, Research Report No. 2615 of the "Institut national de recherche en informatique et en automatique", Rocquencourt, 1995.

[19] C. T. Lawrence, A. L. Tits, *Feasible Sequential Quadratic Programming for Finely Discretized Problems from SIP*, in R. Reemtsen, J.-J. Ruckmann (eds.): Semi-Infinite Programming, in the series Nonconcex Optimization and its Applications. Kluwer Academic Publishers, 1998.

[20] J. L. Zhou, A. L. Tits, *Nonmonotone Line Search for Minimax Problems*, Journal of Optimization Theory and Applications, Vol. 76, No. 3, 1993, pp. 455 - 476.

[21] J. F. Bonnans, E. Panier, A. L. Tits, J. L. Zhou, *Avoiding the Maratos Effect by Means of a Nonmonotone Line search: II. Inequality Problems - Feasible Iterates*, SIAM Journal on Numerical Analysis, Vol. 29, No. 4, 1992, pp. 1187-1202.

[22] C. T. Lawrence, J. L. Zhou, A. L. Tits, *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying all Inequality Constraints*, Institute for Systems Research, University of Maryland, Technical Report TR-94-16r1, 1997.

[23] *The FSQP Home page*, electronic document at http://www.isr.umd.edu/Labs/CACSE/FSQP/fsqp.html , maintained by the Institute for Systems Research, University of Maryland.

[24] H. J. Greenberg, *Mathematical Programming Glossary*, electronic document at http://www.cudenver.edu/~hgreenbe/glossary/glossary.html , 1999.

[25] *Optimization Frequently Asked Questions*, electronic document at http://www-unix.mcs.anl.gov/otc/Guide/faq/ , maintained by Robert Fourer, The Optimization Technology Center.

[26] W.H. Press, S.S. Teukolsky, V.T. Vetterling, B.P. Flannery, *Numerical Recipies in C – the Art of Scientific Computing,* Cambridge University Press, Cambridge, 1992.

[27] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[28] L. N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[29] B. Jacob, *Linear Algebra*, W. H. Freeman and Company, New York, 1990.

[30] I. N. Bronstein, K. A. Smendljajew, G. Musiol, H. Mühlig, *Taschenbuch des Mathematik (second edition - in German),* Verlag Harri Deutsch, Frankfurt am Main, 1995.

[31] I. Kuščer, A. Kodre, H. Neunzert, *Mathematik in Physik und Technik (in German),* Springer - Verlag, Heidelberg, 1993.

[32] E. Kreyszig, *Advanced Engineering Mathematics (second edition)*, John Wiley & Sons, New York, 1993.

[33] Z. Bohte, *Numerične metode*, Društvo matematikov, fizikov in astronomov SRS, Ljubljana, 1987.

[34] K. J. Bathe, *Finite Element Procedures*, p.p. 697-745, Prentice Hall, New Jersey, 1996.

[35] F. van Keulen, V. V. Toropov, *Multipoint Approximations for Structural Optimization Problems with Noisy Response Functions*, electronic document at http://www-tm.wbmt.tudelft.nl/~wbtmavk/issmo/paper/mam_nois2.htm.

[36] J. F. Rodriguez, J. E. Renaud, *Convergence of Trust Region Augmented Lagrangian Methods Using Variable Fidelity Approximation Data*, In: WCSMO-2 : proceedings of the Second World Congress of Structural and Multidisciplinary Optimization, Zakopane, Poland, May 26-30, 1997. Vol. 1, Witold Gutkowski, Zenon Mroz (editors), 1st ed., Lublin, Poland, Wydawnictwo ekoincynieria (WE), 1997, pp. 149-154.