

Università degli Studi di Roma “La Sapienza”
Dipartimento di Informatica e Sistemistica “A. Ruberti”

Gianni Di Pillo and Laura Palagi

**Nonlinear Programming: Introduction,
Unconstrained and Constrained Optimization**

Tech. Rep. 25-01

Dipartimento di Informatica e Sistemistica “A. Ruberti”, Università di Roma “La Sapienza”
via Buonarroti 12 - 00185 Roma, Italy.
E-mail: dipillo@dis.uniroma1.it, palagi@dis.uniroma1.it.

Nonlinear Programming: Introduction, Unconstrained and Constrained Optimization

Gianni Di Pillo and Laura Palagi

Contents

1	Introduction	1
1.1	Problem Definition	1
1.2	Optimality Conditions	2
1.3	Performance of Algorithms	4
1.3.1	Convergence and Rate of Convergence	4
1.3.2	Numerical Behaviour	5
1.4	Selected Bibliography	5
2	Unconstrained Optimization	6
2.1	Line Search Algorithms	7
2.2	Gradient Methods	9
2.3	Conjugate Gradient Methods	9
2.4	Newton's Methods	12
2.4.1	Line Search Modifications of Newton's Method	12
2.4.2	Trust Region Modifications of Newton's Method	14
2.4.3	Truncated Newton's Methods	16
2.5	Quasi-Newton Methods	17
2.6	Derivative Free Methods	19
2.7	Selected Bibliography	21
3	Constrained Optimization	21
3.1	Introduction	21
3.2	Unconstrained Sequential Methods	22
3.2.1	The Quadratic Penalty Method	22
3.2.2	The Logarithmic Barrier Method	23
3.2.3	The Augmented Lagrangian Method	25
3.3	Unconstrained Exact Penalty Methods	27
3.4	Sequential Quadratic Programming Methods	28
3.4.1	The Line Search Approach	32
3.4.2	The Trust Region Approach	32
3.5	Feasible Direction Methods	33
3.6	Selected Bibliography	35

1 Introduction

Nonlinear Programming (NLP) is the broad area of applied mathematics that addresses optimization problems when nonlinearity in the functions are involved. In this chapter we introduce the problem, and making reference to the smooth case, we review the standard optimality conditions that are at the basis of most algorithms for its solution. Then, we give basic notions concerning the performance of algorithms, in terms of convergence, rate of convergence, and numerical behaviour.

1.1 Problem Definition

We consider the problem of determining the value of a vector of *decision variables* $x \in \mathbb{R}^n$ that minimizes an *objective function* $f : \mathbb{R}^n \rightarrow \mathbb{R}$, when x is required to belong to a *feasible set* $\mathcal{F} \subseteq \mathbb{R}^n$; that is we consider the problem:

$$\min_{x \in \mathcal{F}} f(x). \quad (1)$$

Two cases are of main interest:

- the feasible set \mathcal{F} is \mathbb{R}^n , so that Problem (1) becomes:

$$\min_{x \in \mathbb{R}^n} f(x); \quad (2)$$

in this case we say that Problem (1) is *unconstrained*. More in general, Problem (1) is unconstrained if \mathcal{F} is an open set. The optimality conditions for unconstrained problems stated in §1.2 hold also in the general case. Here for simplicity we assume that $\mathcal{F} = \mathbb{R}^n$.

- the feasible set is described by *inequality and/or equality constraints* on the decision variables:

$$\mathcal{F} = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \dots, p; h_j(x) = 0, j = 1, \dots, m\};$$

then Problem (1) becomes:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ & g(x) \leq 0 \\ & h(x) = 0, \end{aligned} \quad (3)$$

with $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$. In this case we say that Problem (1) is *constrained*.

Problem (1) is a *Nonlinear Programming* (NLP) problem when at least one, among the problem functions $f, g_i, i = 1, \dots, p, h_j, j = 1, \dots, m$, is nonlinear in its argument x .

Usually it is assumed that in Problem (3) the number m of equality constraints is not larger than the number n of decision variables. Otherwise the feasible set could be empty, unless there is some dependency among the constraints. If only equality (inequality) constraints are present, Problem (3) is called an *equality (inequality) constrained NLP problem*.

In the following we assume that the problem functions f, g, h are at least *continuously differentiable* in \mathbb{R}^n .

When f is a convex function and \mathcal{F} is a convex set, Problem (1) is a *convex NLP problem*. In particular, \mathcal{F} is convex if the equality constraint functions h_j are affine and the inequality constraint functions g_i are convex. Convexity adds a lot of structure to the NLP problem, and can be exploited widely both from the theoretical and the computational point of view. If f is convex quadratic and h, g are affine, we have a *Quadratic Programming* problem. This is a case of special interest. Here we will confine ourselves to general NLP problems, without convexity assumptions.

A point $x^* \in \mathcal{F}$ is a *global solution* of Problem (1) if $f(x^*) \leq f(x)$, for all $x \in \mathcal{F}$; it is a *strict global solution* if $f(x^*) < f(x)$, for all $x \in \mathcal{F}, x \neq x^*$. A main *existence result* for a

constrained problem is that a global solution exists if \mathcal{F} is compact (Weierstrass Theorem). An easy consequence for unconstrained problems is that a global solution exists if the *level set* $\mathcal{L}_\alpha = \{x \in \mathbb{R}^n : f(x) \leq \alpha\}$ is compact for some finite α .

A point $x^* \in \mathcal{F}$ is a *local solution* of Problem (1) if there exists an open neighborhood \mathcal{B}_{x^*} of x^* such that $f(x^*) \leq f(x)$, for all $x \in \mathcal{F} \cap \mathcal{B}_{x^*}$; it is a *strict local solution* if $f(x^*) < f(x)$, for all $x \in \mathcal{F} \cap \mathcal{B}_{x^*}, x \neq x^*$. Of course, a global solution is also a local solution.

To determine a global solution of a NLP problem is in general a very difficult task. Usually, NLP algorithms are able to determine only local solutions. Nevertheless, in practical applications, also to get a local solution can be of great worth.

We introduce some notation. We denote by the apex $'$, the transpose of a vector or a matrix. Given a function $v : \mathbb{R}^n \rightarrow \mathbb{R}$, we denote by $\nabla v(x)$ the gradient vector and by $\nabla^2 v(x)$ the Hessian matrix of v . Given a vector function $w : \mathbb{R}^n \rightarrow \mathbb{R}^q$, we denote by $\nabla w(x)$ the $n \times q$ matrix whose columns are $\nabla w_j(x), j = 1, \dots, q$. Given a vector $y \in \mathbb{R}^q$ we denote by $\|y\|$ its Euclidean norm. Let $\mathcal{K} \subset \{1, \dots, q\}$ be an index subset, y be a vector with components $y_i, i = 1, \dots, q$, and A be a matrix with columns $a_j, j = 1, \dots, q$. We denote by $y_{\mathcal{K}}$ the subvector of y with components y_i such that $i \in \mathcal{K}$ and by $A_{\mathcal{K}}$ the submatrix of A made up of the columns a_j with $j \in \mathcal{K}$.

1.2 Optimality Conditions

Local solutions must satisfy *necessary optimality conditions* (NOC).

For the unconstrained Problem (2) we have the well know result of classical calculus:

Proposition 1.1 *Let x^* be a local solution of Problem (2), then*

$$\nabla f(x^*) = 0; \tag{4}$$

moreover, if f is twice continuously differentiable, then

$$y' \nabla^2 f(x^*) y \geq 0, \forall y \in \mathbb{R}^n. \tag{5}$$

For the constrained problem (3), most of the NOC commonly used in the development of algorithms assume that at a local solution the constraints satisfy some qualification condition to prevent the occurrence of degenerate cases. These conditions are usually called *constraints qualifications* and among them the *linear independence constraints qualification* (LICQ) is the simplest and by far the most invoked.

Let $\hat{x} \in \mathcal{F}$. We say that the inequality constraint g_i is *active* at \hat{x} if $g_i(\hat{x}) = 0$. We denote by $\mathcal{I}_a(\hat{x})$ the index set of inequality constraints active at \hat{x} :

$$\mathcal{I}_a(\hat{x}) = \{i \in \{1, \dots, p\} : g_i(\hat{x}) = 0\}. \tag{6}$$

Of course, any equality constraint h_j , is active at \hat{x} . LICQ is satisfied at \hat{x} if the gradients of the active constraints $\nabla g_{\mathcal{I}_a}(\hat{x}), \nabla h(\hat{x})$, are linearly independent.

Under LICQ, the NOC for Problem (3) are stated making use of the (generalized) *Lagrangian function*:

$$L(x, \lambda, \mu) = f(x) + \lambda' g(x) + \mu' h(x), \tag{7}$$

where $\lambda \in \mathbb{R}^p, \mu \in \mathbb{R}^m$ are called (generalized Lagrange) *multipliers*, or *dual variables*.

The so called *Karush-Kuhn-Tucker* (KKT) NOC are stated as follows:

Proposition 1.2 *Assume that x^* is a local solution of Problem (3) and that LICQ holds at x^* ; then multipliers $\lambda^* \geq 0, \mu^*$ exist such that:*

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, \mu^*) &= 0, \\ \lambda^{*'} g(x^*) &= 0; \end{aligned} \tag{8}$$

moreover, if f, g, h are twice continuously differentiable, then:

$$y' \nabla_x^2 L(x^*, \lambda^*, \mu^*) y \geq 0, \quad \forall y \in \mathcal{N}(x^*),$$

where:

$$\mathcal{N}(x^*) = \{y \in \mathbb{R}^n : \nabla g_{\mathcal{I}_a}(x^*)' y = 0; \nabla h(x^*)' y = 0\}.$$

A point x^* satisfying the NOC conditions (8) together with some multipliers λ^*, μ^* is called a KKT point.

If a point $x^* \in \mathcal{F}$ satisfies a *sufficient optimality condition*, then it is a local solution of Problem (1). For general NLP problems, sufficient optimality conditions can be stated under the assumption that the problem functions are twice continuously differentiable, so that we have *second order sufficiency conditions* (SOSC).

For the unconstrained Problem (2) we have the SOSC:

Proposition 1.3 *Assume that x^* satisfies the NOC of Proposition (1.1). Assume further that*

$$y' \nabla^2 f(x^*) y > 0, \quad \forall y \in \mathbb{R}^n, y \neq 0,$$

that is that assume that $\nabla^2 f(x^)$ is positive definite. Then x^* is a strict local solution of Problem (2).*

For the constrained Problem (3) we have the SOSC:

Proposition 1.4 *Assume that $x^* \in \mathcal{F}$ and λ^*, μ^* satisfy the NOC of Proposition (1.2). Assume further that*

$$y' \nabla_x^2 L(x^*, \lambda^*, \mu^*) y > 0, \quad \forall y \in \mathcal{P}(x^*), y \neq 0, \quad (9)$$

where:

$$\begin{aligned} \mathcal{P}(x^*) = \{y \in \mathbb{R}^n : & \nabla g_{\mathcal{I}_a}(x^*)' y \leq 0, \quad \nabla h(x^*)' y = 0; \\ & \nabla g_i(x^*)' y = 0, i \in \mathcal{I}_a(x^*) \text{ with } \lambda_i^* > 0\}; \end{aligned}$$

then x^ is a strict local solution of Problem (3).*

Note that $\mathcal{P}(x^*)$ is a polyhedral set, while $\mathcal{N}(x^*)$ is the null space of a linear operator, with $\mathcal{N}(x^*) \subseteq \mathcal{P}(x^*)$. However, if $\lambda_i^* > 0$ for all $i \in \mathcal{I}_a(x^*)$, then $\mathcal{N}(x^*) = \mathcal{P}(x^*)$. When this happens, we say that the *strict complementarity assumption* is satisfied by $g(x^*)$ and λ^* . Strict complementarity is a very favorable circumstance, because it is much simpler to deal with the null space of a linear operator than with a general polyhedron. In particular, there exists a simple algebraic criterion to test whether the quadratic form $y' \nabla_x^2 L(x^*, \lambda^*, \mu^*) y$ is positive definite on $\mathcal{N}(x^*)$.

Note also that, if in Proposition 1.4 we substitute the set $\mathcal{P}(x^*)$ with the set

$$\mathcal{N}^+(x^*) = \{y \in \mathbb{R}^n : \nabla h(x^*)' y = 0, \nabla g_i(x^*)' y = 0, i \in \mathcal{I}_a(x^*) \text{ with } \lambda_i^* > 0\},$$

we still have a sufficient condition, since $\mathcal{P}(x^*) \subseteq \mathcal{N}^+(x^*)$, and again $\mathcal{N}^+(x^*)$ is the null space of a linear operator. However the condition obtained is stronger than the original one, and indeed it is known as the *strong* second order sufficient condition (SSOSC).

Finally, we point out that, under the strict complementarity assumption, $\mathcal{N}(x^*) = \mathcal{P}(x^*) = \mathcal{N}^+(x^*)$, so that all the SOSC for Problem (3) considered above reduce to the same condition.

A main feature of convex problems is that a (strict) local solution is also a (strict) global solution. Moreover, when f is (strictly) convex and, if present, g_i are convex and h_j are affine, the NOC given in terms of first order derivatives are also sufficient for a point x^* to be a (strict) global solution.

Optimality conditions are fundamental in the solution of NLP problems. If it is known that a global solution exists, the most straightforward method to employ them is as follows: find all points satisfying the first order necessary conditions, and declare as global solution the point with the smallest value of the objective function. If the problem functions are twice differentiable, we can also check the second order necessary condition, filtering out those points that do not satisfy it; for the remaining candidates, we can check a second order sufficient condition to find local minima.

It is important to realize, however, that except for very simple cases, using optimality conditions as described above *does not work*. The reason is that, even for an unconstrained problem, to find a solution of the system of equations $\nabla f(x) = 0$ is nontrivial; algorithmically, it is usually as difficult as solving the original minimization problem.

The principal context in which optimality conditions become useful is the development and analysis of algorithms. An algorithm for the solution of Problem (1) produces a sequence $\{x^k\}, k = 0, 1, \dots$, of tentative solutions, and terminates when a stopping criterion is satisfied. Usually the stopping criterion is based on satisfaction of necessary optimality conditions within a prefixed tolerance; moreover, necessary optimality conditions often suggest how to improve the current tentative solution x^k in order to get the next one x^{k+1} , closer to the optimal solution. Thus, necessary optimality conditions provide the basis for the convergence analysis of algorithms. On the other hand, sufficient optimality conditions play a key role in the analysis of the rate of convergence.

1.3 Performance of Algorithms

1.3.1 Convergence and Rate of Convergence

Let $\Omega \subset \mathcal{F}$ be the subset of points that satisfy the first order NOC for Problem (1). From a theoretical point of view, an optimization algorithm stops when a point $x^* \in \Omega$ is reached. From this point of view, the set Ω is called the *target set*. Convergence properties are stated with reference to the target set Ω . In the unconstrained case, a possible target set is $\Omega = \{x \in \mathbb{R}^n : \nabla f(x) = 0\}$ whereas in the constrained case Ω may be the set of KKT points satisfying (8).

Let $\{x^k\}, k = 0, 1, \dots$ be the sequence of points produced by an algorithm. Then, the algorithm is *globally convergent* if a limit point x^* of $\{x^k\}$ exists such that $x^* \in \Omega$ for any starting point $x^0 \in \mathbb{R}^n$; it is *locally convergent* if the existence of the limit point $x^* \in \Omega$ can be established only if the starting point x^0 belongs to some neighborhood of Ω .

The notion of convergence stated before is the weakest that ensures that a point x^k arbitrarily close to Ω can be obtained for k large enough. In the unconstrained case this implies that

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0.$$

Nevertheless, stronger convergence properties can often be established. For instance that any subsequence of $\{x^k\}$ possess a limit point, and any limit point of $\{x^k\}$ belongs to Ω ; for the unconstrained case, this means that

$$\lim_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0.$$

The strongest convergence requirement is that the whole sequence $\{x^k\}$ converges to a point $x^* \in \Omega$.

In order to define the rate of convergence of an algorithm, it is assumed for simplicity that the algorithm produces a sequence $\{x^k\}$ converging to a point $x^* \in \Omega$. The most widely employed notion of rate of convergence is the *Q-rate of convergence*, that considers the quotient

between two successive iterates given by $\|x^{k+1} - x^*\|/\|x^k - x^*\|$. Then we say that the rate of convergence is *Q-linear* if there is a constant $r \in (0, 1)$ such that:

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} \leq r, \text{ for all } k \text{ sufficiently large};$$

we say that the rate of convergence is *Q-superlinear* if:

$$\lim_{k \rightarrow \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = 0;$$

and we say that the rate of convergence is *Q-quadratic* if:

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} \leq R, \text{ for all } k \text{ sufficiently large},$$

where R is a positive constant, not necessarily less than 1. More in general, the rate of convergence is of *Q-order* p if there exists a positive constant R such that:

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^p} \leq R, \text{ for all } k \text{ sufficiently large};$$

however a Q-order larger than 2 is very seldom achieved. Algorithms of common use are either superlinearly or quadratically convergent.

1.3.2 Numerical Behaviour

Besides the theoretical performance, an important aspect of a numerical algorithm is its practical performance. Indeed fast convergence rate can be overcome by a great amount of algebraic operations needed at each iteration. Different measures of the numerical performance exist, even if direct comparison of CPU time remains the best way of verifying the efficiency of a given method on a specific problem. However, when the computational burden in term of algebraic operations per iteration is comparable, possible measures of performance can be given in terms of number of *iterations*, number of *objective function/constraints evaluations* and number of *its/their gradient evaluations*.

Measures of performance are of main relevance for *large scale* NLP problems. The notion of large scale is machine dependent so that it could be difficult to state a priori when a problem is large. Usually, an unconstrained problem with $n \geq 1000$ is considered large whereas a constrained problems without particular structure is considered large when $n \geq 100$ and $p+m \geq 100$. A basic feature of an algorithm for large scale problems is a low storage overhead needed to make practicable its implementation, and a measure of performance is usually the *number of matrix-vector products* required. The main difficulty in dealing with large scale problems is that effective algorithms for small scale problems do not necessarily translate into efficient ones when applied to solve large problems.

1.4 Selected Bibliography

In this section we give a list of books and surveys of general references in NLP. Of course the list is by far not exhaustive; it includes only texts widely employed and currently found.

Several textbooks on applied optimization devote chapters to NLP theory and practice. Among them, the books by Gill et al. (1981), Minoux (1983), Luenberger (1994), Fletcher (1987), Nash and Sofer (1996) Bonnans et al. (1997) and Nocedal and Wright (1999).

More specialized textbooks, strictly devoted to NLP, are the ones by Zangwill (1969), Avriel (1976), McCormick (1983), Evtushenko (1985), Bazaraa et al. (1993), Polak (1997), Bertsekas (1999). A collection of survey papers on algorithmic aspects of NLP is in Spedicato (1994).

A particular emphasis on the mathematical foundations of NLP is given in Hestenes (1975), Giannessi (1982), Polyak (1987), Peressini et al. (1988), Jahn (1994), Rapsák (1997). A classical reference on optimality conditions and constraint qualifications is the book by Mangasarian (1969).

The theoretical analysis of algorithms for NLP is developed in Zangwill (1969), Orthega and Rheinboldt (1970), Bazaraa et al. (1993), Luenberger (1994), Polak (1997). For practical aspects in the implementation of algorithms we refer again to Gill et al. (1981).

The search for global optima is treated for instance in Törn and Žilinskas (1989), Horst and Pardalos (1995), Pinter (1996).

This chapter does not mention *multiobjective* optimization. Multiobjective NLP problems are considered for instance in Rustem (1998), Miettinen (1999).

2 Unconstrained Optimization

In this chapter we consider algorithms for solving the *unconstrained* Nonlinear Programming problem (2)

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

where $x \in \mathbb{R}^n$ is the vector of decision variables and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function. The algorithms described in this section can be easily adapted to solve the constrained Problem $\min_{x \in \mathcal{F}} f(x)$ when \mathcal{F} is an open set, provided that a feasible point $x^0 \in \mathcal{F}$ is available.

In the following we assume for simplicity that the problem function f is twice continuously differentiable in \mathbb{R}^n even if in many cases only once continuously differentiability is needed.

We also assume that the standard assumption ensuring the existence of a solution of Problem (2) holds, namely that:

Assumption 2.1 *The level set $\mathcal{L}^0 = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ is compact, for some $x^0 \in \mathbb{R}^n$.*

The algorithm models treated in this section generate a sequence $\{x^k\}$, starting from x^0 , by the following iteration

$$x^{k+1} = x^k + \alpha^k d^k, \tag{10}$$

where d^k is a *search direction* and α^k is a *stepsize* along d^k .

Methods differentiate in the way the direction and the stepsize are chosen. Of course different choices of d^k and α^k yield different convergence properties. Roughly speaking, the search direction affects the local behaviour of an algorithm and its rate of convergence whereas global convergence is often tied to the choice of the stepsize α^k . The following proposition states a rather general convergence result (Orthega 1988).

Proposition 2.2 *Let $\{x^k\}$ be the sequence generated by iteration (10). Assume that: (i) $d^k \neq 0$ if $\nabla f(x^k) \neq 0$; (ii) for all k we have $f(x^{k+1}) \leq f(x^k)$; (iii) we have*

$$\lim_{k \rightarrow \infty} \frac{\nabla f(x^k)' d^k}{\|d^k\|} = 0; \tag{11}$$

(iv) for all k with $d^k \neq 0$ we have

$$\frac{|\nabla f(x^k)' d^k|}{\|d^k\|} \geq c \|\nabla f(x^k)\|, \text{ with } c > 0.$$

Then, either \bar{k} exists such that $x^{\bar{k}} \in \mathcal{L}^0$ and $\nabla f(x^{\bar{k}}) = 0$, or an infinite sequence is produced such that:

- (a) $\{x^k\} \in \mathcal{L}^0$;
- (b) $\{f(x^k)\}$ converges;
- (c)

$$\lim_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0. \quad (12)$$

Condition (c) of Proposition 2.2 corresponds to say that any subsequence of $\{x^k\}$ possess a limit point, and any limit point of $\{x^k\}$ belongs to Ω . However, weaker convergence results can be proved (see Section 1). In particular in some cases, only condition

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0 \quad (13)$$

holds.

Many methods are classified according to the information they use regarding the smoothness of the function. In particular we will briefly describe gradient methods, conjugate gradient methods, Newton's and truncated Newton's methods, quasi-Newton methods and derivative free methods.

Most methods determine α^k by means of a line search technique. Hence we start with a short review of the most used techniques and more recent developments in *line search algorithms*.

2.1 Line Search Algorithms

Line Search Algorithms (LSA) determine the stepsize α^k along the direction d^k . The aim of different choices of α^k is mainly to ensure that the algorithm defined by (10) results to be globally convergent, possibly without deteriorating the rate of convergence. A first possibility is to set $\alpha^k = \alpha^*$ with

$$\alpha^* = \arg \min_{\alpha} f(x^k + \alpha d^k),$$

that is α^k is the value that minimizes the function f along the direction d^k . However, unless the function f has some special structure (being quadratic), an exact line search is usually quite computationally costly and its use may not be worthwhile. Hence approximate methods are used. In the cases when ∇f can be computed, we assume that the condition

$$\nabla f(x^k)' d^k < 0, \quad \text{for all } k \quad (14)$$

holds, namely we assume that d^k is a *descent direction* for f at x^k .

Among the simplest LSAs is *Armijo's method*, reported in Algorithm 1.

The choice of the initial step Δ^k is tied to the direction d^k used. By choosing Δ^k as in Step 1, it is possible to prove that Armijo's LSA finds in a finite number of steps a value α^k such that

$$f(x^{k+1}) < f(x^k), \quad (15)$$

and that condition (11) holds.

Armijo's LSA finds a stepsize α^k that satisfies a condition of sufficient decrease of the objective function and implicitly a condition of sufficient displacement from the current point x^k .

In many cases it can be necessary to impose stronger conditions on the stepsize α^k . In particular *Wolfe's conditions* are widely employed. In this case, Armijo's condition

$$f(x^k + \alpha d^k) \leq f(x^k) + \gamma \alpha \nabla f(x^k)' d^k \quad (16)$$

is coupled either with condition

$$\nabla f(x^k + \alpha d^k)' d^k \geq \beta \nabla f(x^k)' d^k, \quad (17)$$

Armijo's LSA**Data:** $\delta \in (0, 1)$, $\gamma \in (0, 1/2)$, $c \in (0, 1)$.**Step 1.** Choose Δ^k such that

$$\Delta^k \geq c \frac{|\nabla f(x^k)' d^k|}{\|d^k\|^2}$$

and set $\alpha = \Delta^k$.**Step 2.** If

$$f(x^k + \alpha d^k) \leq f(x^k) + \gamma \alpha \nabla f(x^k)' d^k$$

then $\alpha^k = \alpha$ and stop.**Step 3.** Otherwise set $\alpha = \delta \alpha$ and go to Step 2.

Table 1: Armijo's line search algorithm

or with the stronger one

$$|\nabla f(x^k + \alpha^k d^k)' d^k| \leq \beta |\nabla f(x^k)' d^k|, \quad (18)$$

where $\beta \in (\gamma, 1)$, being γ the value used in (16). It is possible to prove that a finite interval of values of α exists where conditions (16) and (18) are satisfied (and hence also (16) and (17)). Of course, a LSA satisfying (16) and (17) enforces conditions (11) and (15). LSA for finding a stepsize α^k that satisfies Wolfe's conditions are more complicated than Armijo's LSA and we do not enter details.

In some cases, such as for derivative free methods, it can be necessary to consider LSA that do not require the use of derivatives. In this case, condition (14) cannot be verified and a direction d^k cannot be ensured to be of descent. Hence line search techniques must take into account the possibility of using also a negative stepsize, that is to move along the opposite direction $-d^k$. Moreover, suitable conditions for terminating the line search with $\alpha^k = 0$ must be imposed when it is likely that $f(x^k + \alpha d^k)$ has a minimizer at $\alpha = 0$. A possible LSA is obtained by substituting the Armijo acceptance rule (16) by a condition of the type

$$f(x^k + \alpha d^k) \leq f(x^k) - \gamma \alpha^2 \|d^k\|^2, \quad (19)$$

that does not require gradient information. We point out that derivative free LSAs usually guarantee the satisfaction of the condition $\lim_{k \rightarrow \infty} \|x^{k+1} - x^k\| = 0$, that can be useful to prove convergence whenever it is not possible to control the angle between the gradient $\nabla f(x^k)$ and the direction d^k . See De Leone et al. (1984), Grippo et al. (1988) for more sophisticated derivative free line search algorithms.

We observe that all the line search algorithms described so far ensure that condition (15) is satisfied, namely they ensure a monotonic decrease in the objective function values. Actually, enforcing monotonicity may not be necessary to prove convergence results and, in case of highly nonlinear functions, may cause minimization algorithms to be trapped within some narrow region. A very simple *nonmonotone line search* scheme (Grippo et al. 1986) consists in relaxing Armijo's condition (16), requiring that the function value in the new iterate satisfies an Armijo type rule with respect to a prefixed number of previous iterates, namely that

$$f(x^k + \alpha d^k) \leq \max_{0 \leq j \leq J} \{f(x^{k-j})\} + \gamma \alpha \nabla f(x^k)' d^k, \quad (20)$$

where J is the number of previous iterates that are taken into account. Numerical experiences have proved the efficiency of the use of nonmonotone schemes, see e.g. Ferris et al. (1996), Raydan (1997), Lucidi et al. (1998b).

2.2 Gradient Methods

The gradient method, also known as the *steepest descent method*, is considered a basic method in unconstrained optimization and is obtained by setting $d^k = -\nabla f(x^k)$ in the algorithm (10). It requires only information about first order derivatives and hence is very attractive because of its limited computational cost and storage requirement. Moreover, it is considered the most significant example of globally convergent algorithm, in the sense that using a suitable LSA it is possible to establish a global convergence result. However, its rate of convergence is usually very poor and hence it is not widely used on its own, even if recent developments have produced more efficient versions of the method. A possible algorithmic scheme is reported in Algorithm 2.

The Gradient Algorithm	
Step 1.	Choose $x^0 \in R^n$ and set $k = 0$.
Step 2.	If $\nabla f(x^k) = 0$ stop.
Step 3.	Set $d^k = -\nabla f(x^k)$ and find α^k by using the Armijo's LSA of Algorithm 1.
Step 4.	Set
	$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$
	$k = k + 1$ and go to Step 2.

Table 2: The gradient method

The choice of the initial stepsize Δ^k in the Armijo's LSA is critical and can affect significantly the behaviour of the gradient method. As regards the convergence of the method, Proposition 2.2 can be easily adapted; nonetheless we have also the following result that holds even without requiring the compactness of the level set \mathcal{L}^0 .

Proposition 2.3 *If ∇f is Lipschitz continuous and f is bounded from below, then the sequence generated by the gradient method of Algorithm 2 satisfies condition (12).*

Hence the gradient method is satisfactory as concerns global convergence. However, only a linear convergence rate can be proved. Indeed also from a practical point of view, the rate of convergence is usually very poor and it strongly depends on the condition number of the Hessian matrix $\nabla^2 f$.

Barzilai and Borwein (1988) proposed a particular rule for computing the stepsize in gradient methods that leads to a significant improvement in numerical performance. In particular the stepsize is chosen as

$$\alpha^k = \frac{\|x^k - x^{k-1}\|^2}{(x^k - x^{k-1})'(\nabla f(x^k) - \nabla f(x^{k-1}))}. \quad (21)$$

In Raydan (1997) a globalization strategy has been proposed that tries to accept the stepsize given by (21) as frequently as possible. The globalization strategy is based on a Armijo's nonmonotone line search of type (20). Condition (12) can still be proved. Moreover, the numerical experience shows that this method is competitive also with some version of conjugate gradient methods.

2.3 Conjugate Gradient Methods

The conjugate gradient method is very popular due to its simplicity and low computational requirements. The method was originally introduced for solving the minimization of a strictly

convex quadratic function with the aim of accelerating the gradient method without requiring the overhead needed by the Newton's method (see §2.4). Indeed it requires only first order derivatives and no matrix storage and operations are needed.

The basic idea is that the minimization on \mathbb{R}^n of the strictly convex quadratic function

$$f(x) = \frac{1}{2}x'Qx + a'x$$

with Q symmetric positive definite, can be split into n minimizations over \mathbb{R} . This is done by means of n directions d^0, \dots, d^{n-1} *conjugate* with respect to the Hessian Q , that is directions such that $(d^j)'Qd^i = 0$ for $j \neq i$. Along each direction d^j an exact line search is performed in closed form. This corresponds to the *conjugate directions algorithm* (CDA) that finds the global minimizer of a strictly convex quadratic function in at most n iterations and that is reported in Algorithm 3. The value of α^k at Step 3 is the exact minimizer of the quadratic function $f(x^k + \alpha d^k)$. Note that CDA can be seen equivalently as an algorithm for solving $\nabla f(x) = 0$, that is for solving the linear system $Qx = -a$.

CDA for quadratic functions	
Data.	Q -conjugate directions d^0, \dots, d^{n-1} .
Step 1.	Choose $x^0 \in R^n$ and set $k = 0$.
Step 2.	If $\nabla f(x^k) = 0$ stop.
Step 3.	Set
	$\alpha^k = \frac{\nabla f(x^k)'d^k}{(d^k)'Qd^k}.$
Step 4.	Set
	$x^{k+1} = x^k + \alpha^k d^k,$
	$k = k + 1$ and go to Step 2.

Table 3: Conjugate directions algorithm for quadratic functions

In the CDA, the conjugate directions are given, whereas in the *conjugate gradient algorithm* (CGA) the n conjugate directions are generated iteratively according to the rule

$$d^k = \begin{cases} -\nabla f(x^k) & k = 0 \\ -\nabla f(x^k) + \beta^{k-1}d^{k-1} & k \geq 1. \end{cases}$$

The scalar β^k is chosen in order to enforce conjugacy among the directions. The most common choices for β^k are either the Fletcher-Reeves formula (Fletcher and Reeves 1964)

$$\beta_{FR}^k = \frac{\|\nabla f(x^{k+1})\|^2}{\|\nabla f(x^k)\|^2}, \tag{22}$$

or the Polak-Ribière formula (Polak and Ribière 1969)

$$\beta_{PR}^k = \frac{\nabla f(x^{k+1})'(\nabla f(x^{k+1}) - \nabla f(x^k))}{\|\nabla f(x^k)\|^2}. \tag{23}$$

The two formulas (22) and (23) are equivalent in the case of quadratic objective function.

When f is not a quadratic function the two formulas (22) and (23) give different values for β and an inexact line search is performed to determine the stepsize α^k . Usually a LSA that enforces the strong Wolfe's conditions (16) and (18) is used. A scheme of CGA is reported

CGA for nonlinear functions

Step 1. Choose $x^0 \in R^n$ and set $k = 0$.

Step 2. If $\nabla f(x^k) = 0$ stop.

Step 3. Compute β^{k-1} by either (22) or (23) and set the direction

$$d^k = \begin{cases} -\nabla f(x^k) & k = 0 \\ -\nabla f(x^k) + \beta^{k-1}d^{k-1} & k \geq 1; \end{cases}$$

Step 4. Find α^k by means of a LSA that satisfies the strong Wolfe's conditions (16) and (18).

Step 5. Set

$$x^{k+1} = x^k + \alpha^k d^k,$$

$k = k + 1$ and go to Step 2.

Table 4: Conjugate gradient algorithm for nonlinear functions

in Algorithm 4. The simplest device used to guarantee global convergence properties of CG methods is a periodic *restart* along the steepest descent direction. Restarting can also be used to deal with the loss of conjugacy due to the presence of nonquadratic terms in the objective function that may cause the method to generate inefficient or even meaningless directions. Usually the restart procedure occurs every n steps or if a test on the loss of conjugacy such as

$$|\nabla f(x^k)' \nabla f(x^{k-1})| > \delta \|\nabla f(x^{k-1})\|^2,$$

with $0 < \delta < 1$, is satisfied.

However, in practice, the use of a regular restart may not be the most convenient technique for enforcing global convergence. Indeed CGA are used for large scale problems when n is very large and the expectation is to solve the problem in less than n iterations, that is before restart occurs. Without restart, the CGA with $\beta^k = \beta_{FR}^k$ and a LSA that guarantees the satisfaction of the strong Wolfe's conditions, is globally convergent in the sense that (13) holds (Al-Baali 1985). Actually in Gilbert and Nocedal (1992) the same type of convergence has been proved for any value of β^k such that $0 < |\beta^k| \leq \beta_{FR}^k$. Analogous convergence results do not hold for the Polak-Ribière formula even if the numerical performance of this formula is usually superior to Fletcher-Reeves one. This motivated a big effort to find a globally convergent version of the Polak-Ribière method. Indeed condition (13) can be established for a modification of the Polak-Ribière CG method that uses only positive values of $\beta^k = \max\{\beta_{PR}^k, 0\}$ and a LSA that ensures satisfaction of the strong Wolfe's conditions and of the additional condition

$$\nabla f(x^k)' d^k \leq -c \|\nabla f(x^k)\|^2$$

for some constant $c > 0$ (Powell 1986).

Recently it has been proved that the Polak-Ribière CG method satisfies the stronger convergence condition (12) if a more sophisticated line search technique is used (Grippo and Lucidi 1997). Indeed a LSA that enforces condition (19) coupled with a sufficient descent condition of the type

$$-\delta_2 \|\nabla f(x^{k+1})\|^2 \leq \nabla f(x^{k+1})' d^{k+1} \leq -\delta_1 \|\nabla f(x^{k+1})\|^2$$

with $0 < \delta_1 < 1 < \delta_2$ must be used.

With respect to the numerical performance, the CGA is also affected by the condition number of the Hessian $\nabla^2 f$. This leads to the use of *preconditioned conjugate gradient methods* and many papers have been devoted to the study of efficient preconditioners.

2.4 Newton's Methods

The Newton's method is considered one of the most powerful algorithms for solving unconstrained optimization problems (see (Moré and Sorensen 1984) for a review). It relies on the quadratic approximation

$$q^k(s) = \frac{1}{2}s'\nabla^2 f(x^k)s + \nabla f(x^k)'s + f(x^k)$$

of the objective function f in the neighborhood of the current iterate x^k and requires the computation of ∇f and $\nabla^2 f$ at each iteration k . Newton's direction d_N^k is obtained as a stationary point of the quadratic approximation q^k , that is as a solution of the system:

$$\nabla^2 f(x^k)d_N = -\nabla f(x^k). \quad (24)$$

Provided that $\nabla^2 f(x^k)$ is non-singular, Newton's direction is given by $d_N^k = -[\nabla^2 f(x^k)]^{-1}\nabla f(x^k)$ and the basic algorithmic scheme is defined by the iteration

$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1}\nabla f(x^k); \quad (25)$$

we refer to (25) as the *pure Newton's method*. The reputation of Newton's method is due to the fact that, if the starting point x^0 is close enough to a solution x^* , then the sequence generated by iteration (25) converges to x^* superlinearly (quadratically if the Hessian $\nabla^2 f$ is Lipschitz continuous in a neighborhood of the solution).

However, the pure Newton's method presents some drawbacks. Indeed $\nabla^2 f(x^k)$ may be singular, and hence Newton's direction cannot be defined; the starting point x^0 can be such that the sequence $\{x^k\}$ generated by iteration (25) does not converge and even convergence to a maximum point may occur. Therefore, the pure Newton's method requires some modification that enforces the global convergence to a solution, preserving the rate of convergence. A *globally convergent Newton's method* must produce a sequence $\{x^k\}$ such that: (i) it admits a limit point; (ii) any limit point belongs to the level set \mathcal{L}^0 and is a stationary point of f ; (iii) no limit point is a maximum point of f ; (iv) if x^* is a limit point of $\{x^k\}$ and $\nabla^2 f(x^*)$ is positive definite, then the convergence rate is at least superlinear.

There are two main approaches for designing globally convergent Newton's method, namely the *line search approach* and the *trust region approach*. We briefly describe them in the next two sections.

2.4.1 Line Search Modifications of Newton's Method

In the line search approach, the pure Newton's method is modified by controlling the magnitude of the step along the Newton's direction d_N^k ; the basic iteration (25) becomes

$$x^{k+1} = x^k - \alpha^k [\nabla^2 f(x^k)]^{-1}\nabla f(x^k), \quad (26)$$

where α^k is chosen by a suitable LSA, for example the Armijo's LSA with initial estimate $\Delta^k = 1$. Moreover, the Newton's direction d_N^k must be perturbed in order to ensure that a globally convergent algorithm is obtained. The simplest way of modifying the Newton's method consists in using the steepest descent direction (possibly after positive diagonal scaling) whenever d_N^k does not satisfy some convergence conditions.

A possible scheme of a globally convergent Newton's algorithm is in Algorithm 5.

We observe that at Step 3, the steepest descent direction is taken whenever $\nabla f(x^k)'d_N^k \geq 0$. Another modification of the Newton's methods consists in taking $d^k = -d_N^k$ if it satisfies $|\nabla f(x^k)'d^k| \geq c_1 \|\nabla f(x^k)\|^q$ and $\|d^k\|^p \leq c_2 \|\nabla f(x^k)\|$. This corresponds to the use of a negative curvature direction d^k that may result in practical advantages.

Line search based Newton's method**Data.** $c_1 > 0, c_2 > 0, p \geq 2, q \geq 3$.**Step 1.** Choose $x_0 \in \mathbb{R}^n$ and set $k = 0$.**Step 2.** If $\nabla f(x^k) = 0$ stop.**Step 3.** If a solution d_N^k of the system

$$\nabla^2 f(x^k) d_N = -\nabla f(x^k)$$

exists and satisfies the conditions

$$\nabla f(x^k)' d_N^k \leq -c_1 \|\nabla f(x^k)\|^q, \quad \|d_N^k\|^p \leq c_2 \|\nabla f(x^k)\|$$

then set the direction $d^k = d_N^k$; otherwise set $d^k = -\nabla f(x^k)$.**Step 4.** Find α^k by using the Armijo's LSA of Algorithm 1.**Step 5.** Set

$$x^{k+1} = x^k + \alpha^k d^k,$$

 $k = k + 1$ and go to Step 2.

Table 5: Line search based Newton's method

The use of the steepest descent direction is not the only possibility to construct globally convergent modifications of Newton's method. Another way is that of perturbing the Hessian matrix $\nabla^2 f(x^k)$ by a diagonal matrix Y^k so that the matrix $\nabla^2 f(x^k) + Y^k$ is positive definite and a solution d of the system

$$(\nabla^2 f(x^k) + Y^k) d = -\nabla f(x^k) \tag{27}$$

exists and provides a descent direction. A way to construct such perturbation is to use a *modified Cholesky factorization* (see e.g. Gill et al. (1981), Lin and Moré (1999)).

The most common versions of globally convergent Newton's methods are based on the enforcement of a monotonic decrease of the objective function values, although neither strong theoretical reasons nor computational evidence support this requirement. Indeed, the convergence region of Newton's method is often much larger than one would expect, but a convergent sequence in this region does not necessarily correspond to a monotonically decreasing sequence of function values. It is possible to relax the standard line search conditions in a way that preserves a global convergence property. More specifically in Grippo et al. (1986) a modification of the Newton's method that uses the Armijo-type nonmonotone rule (20) has been proposed. The algorithmic scheme remains essentially the same of Algorithm 5 except for Step 4. It is reported in Algorithm 6.

Actually, much more sophisticated nonmonotone stabilization schemes have been proposed (Grippo et al. 1991). We do not go into detail about these schemes. However, it is worthwhile to remark that, at each iteration of these schemes only the magnitude of $\|d^k\|$ is checked and if the test is satisfied, the stepsize one is accepted without evaluating the objective function. Nevertheless, a test on the decrease of the values of f is performed at least every M steps.

The line search based Newton's methods described so far converge in general to points satisfying only the first order NOC for Problem (2). This is essentially due to the fact that Newton's method does not exploit all the information contained in the second derivatives. Indeed stronger convergence results can be obtained by using the pair of directions (d^k, s^k) and using a *curvilinear line search*, that is a line search along the curvilinear path

$$x^{k+1} = x^k + \alpha^k d^k + (\alpha^k)^{1/2} s^k, \tag{28}$$

Nonmonotone Newton's method

Data. $c_1 > 0, c_2 > 0, p \geq 2, q \geq 3$ and an integer M .

Step 1. Choose $x_0 \in \mathbb{R}^n$. Set $k = 0, m(k) = 0$.

Step 2. If $\nabla f(x^k) = 0$ stop.

Step 3. If a solution d_N^k of the system

$$\nabla^2 f(x^k) d_N = -\nabla f(x^k)$$

exists and satisfies the conditions

$$\nabla f(x^k)' d_N^k \leq -c_1 \|\nabla f(x^k)\|^q, \quad \|d_N^k\|^p \leq c_2 \|\nabla f(x^k)\|$$

then set the direction $d^k = d_N^k$; otherwise set $d^k = -\nabla f(x^k)$.

Step 4. Find α^k by means of a LSA that enforces the nonmonotone Armjio's rule (20) with $J = m(k)$.

Step 5. Set

$$x^{k+1} = x^k + \alpha^k d^k,$$

$k = k + 1, m(k) = \min\{m(k-1) + 1, M\}$ and go to Step 2.

Table 6: Nonmonotone Newton's method

where d^k is a Newton-type direction and s^k is a particular direction that includes negative curvature information with respect to $\nabla^2 f(x^k)$. By using a suitable modification of the Armjio's rule for the stepsize α^k , algorithms which are globally convergent towards points which satisfy the second order NOC for Problem (2) can be defined both for the monotone (Moré and Sorensen 1979) and the nonmonotone versions (Ferris et al. 1996).

2.4.2 Trust Region Modifications of Newton's Method

In trust region modifications of the Newton's method, the iteration becomes

$$x^{k+1} = x^k + s^k,$$

where the step s^k is obtained by minimizing the quadratic model q^k of the objective function not on the whole space \mathbb{R}^n but on a suitable "trust region" where the model is supposed to be reliable. The trust region is usually defined as an ℓ_p -norm of the step s . The most common choice is the Euclidean norm, so that at each iteration k , the step s^k is obtained by solving

$$\min_{s \in \mathbb{R}^n} \frac{1}{2} s' \nabla^2 f(x^k) s + \nabla f(x^k)' s \quad (29)$$

$$\|s\|^2 \leq (a^k)^2,$$

where a^k is the trust region radius. Often a scaling matrix D^k is used and the spherical constraint becomes $\|D^k s\|^2 \leq (a^k)^2$; this can be seen as a kind of preconditioning. However, by rescaling and without loss of generality, we can assume, for sake of simplicity, that $D^k = I$ where I we denote the identity matrix of suitable dimension.

In trust region schemes the radius a^k is iteratively updated. The idea is that when $\nabla^2 f(x^k)$ is positive definite then the radius a^k should be large enough so that the minimizer of Problem (29) is unconstrained and the full Newton step is taken. The updating rule of a^k depends on the ratio ρ^k between the actual reduction $f(x^k) - f(x^{k+1})$ and the expected reduction $f(x^k) - q^k(s^k)$. A possible scheme is described in Algorithm 7 where the condition at Step 3 is equivalent to the satisfaction of the NOC for Problem (2).

A trust region method

Data. $0 < \gamma_1 \leq \gamma_2 < 1$, $0 < \delta_1 < 1 \leq \delta_2$.

Step 1. Choose $x_0 \in R^n$ and a radius $a^0 > 0$. Set $k = 0$.

Step 2. Find $s^k = \arg \min_{\|s\| \leq a^k} q^k(s)$

Step 3. If $f(x^k) = q^k(s^k)$ stop.

Step 4. Compute the ratio

$$\rho^k = \frac{f(x^k) - f(x^k + s^k)}{f(x^k) - q^k(s^k)}$$

If $\rho^k \geq \gamma_1$ set $x^{k+1} = x^k + s^k$, otherwise set $x^{k+1} = x^k$.

Step 5. Update the radius a^k

$$a^{k+1} = \begin{cases} \delta_1 a^k & \text{if } \rho^k < \gamma_1 \\ a^k & \text{if } \rho^k \in [\gamma_1, \gamma_2] \\ \delta_2 a^k & \text{if } \rho^k > \gamma_2 \end{cases}$$

Set $k = k + 1$ and go to Step 2.

Table 7: Trust region based Newton's method

If f is twice continuously differentiable, there exists a limit point of the sequence $\{x^k\}$ produced by Algorithm 7, that satisfies the first and second order NOC for Problem (2). Furthermore, if x^k converges to a point where the Hessian $\nabla^2 f$ is positive definite, then the rate of convergence is superlinear.

This approach was first proposed in (Sorensen 1982, Moré and Sorensen 1983) and it relies on the fact that Problem (29) always admits a global solution and that a characterization of a global minimizer exists even in the case where the Hessian $\nabla^2 f(x^k)$ is not positive definite. Indeed, the optimal solution s^k satisfies a system of the form

$$[\nabla^2 f(x^k) + \lambda^k I]s^k = -\nabla f(x^k), \quad (30)$$

where λ^k is a KKT multiplier for Problem (29) such that $\nabla^2 f(x^k) + \lambda^k I$ is positive semidefinite. Hence, the trust region modification of Newton's method also fits in the framework of using a correction of the Hessian matrix (27).

The main computational effort in the scheme of Algorithm 7 is the solution of the trust region subproblem at Step 2. The peculiarities of Problem (29) led to the development of *ad hoc* algorithms for finding a global solution. The first ones proposed in literature (Gay 1981, Sorensen 1982, Moré and Sorensen 1983) were essentially based on the solution of a sequence of linear systems of the type (30) for a sequence $\{\lambda^k\}$. These algorithms produce an approximate global minimizer of the trust region subproblem, but rely on the ability to compute at each iteration k a Cholesky factorization of the matrix $(\nabla^2 f(x^k) + \lambda^k I)$. Hence they are appropriate when forming a factorization for different values of λ^k is realistic in terms of both computer storage and time requirements. However, for large scale problems without special structure, one cannot rely on factorizations of the matrices involved, even if, recently, incomplete Cholesky factorizations for large scale problems have been proposed (Lin and Moré 1999). Recent papers concentrate on iterative methods of conjugate gradient type that require only matrix-vector products. Different approaches have been proposed (Pham Dinh and Hoai An 1998, Lucidi et al. 1998a, Rendl and Wolkowicz 1997, Sorensen 1997, Gould et al. 1999).

Actually, the exact solution of Problem (29) is not required. Indeed, in order to prove a

global convergence result of a trust region scheme, it suffices that the model value $q^k(s^k)$ is not larger than the model value at the *Cauchy point*, which is the minimizer of the quadratic model within the trust region along the (preconditioned) steepest-descent direction (Powell 1975). Hence any descent method that moves from the Cauchy point ensures convergence (Toint 1981, Steihaug 1983).

2.4.3 Truncated Newton's Methods

Newton's method requires at each iteration k the solution of a system of linear equations. In the large scale setting, solving exactly the Newton's system can be too burdensome and storing or factoring the full Hessian matrix can be difficult when not impossible. Moreover the exact solution when x^k is far from a solution and $\|\nabla f(x^k)\|$ is large may be unnecessary. On the basis of these remarks, *inexact Newton's methods* have been proposed (Dembo et al. 1982) that approximately solve the Newton system (24) and still retain a good convergence rate. If \tilde{d}_N^k is any approximate solution of the Newton system (24), the measure of accuracy is given by the *residual* r^k of the Newton equation, that is

$$r^k = \nabla^2 f(x^k) \tilde{d}_N^k + \nabla f(x^k).$$

By controlling the magnitude of r^k , superlinear convergence can be proved: if $\{x^k\}$ converges to a solution and if

$$\lim_{k \rightarrow \infty} \frac{\|r^k\|}{\|\nabla f(x^k)\|} = 0, \quad (31)$$

then $\{x^k\}$ converges superlinearly. This result is at the basis of *Truncated Newton's methods* (see Nash (1999) for a recent survey).

Truncated Newton's Algorithm (TNA)	
Data.	k, H, g and a scalar $\eta > 0$.
Step 1.	Set $\varepsilon = \eta \ g\ \min \left\{ \frac{1}{k+1}, \ g\ \right\}$, $i = 0$. $p^0 = 0, r^0 = -g, s^0 = r^0$,
Step 2.	If $(s^i)' H s^i \leq 0$, set $\tilde{d}_N = \begin{cases} -g & \text{if } i = 0 \\ p^i & \text{if } i > 0 \end{cases}$ and stop.
Step 3.	Compute $\alpha^i = \frac{(s^i)' r^i}{(s^i)' H s^i}$ and set $\begin{aligned} p^{i+1} &= p^i + \alpha^i s^i, \\ r^{i+1} &= r^i - \alpha^i H s^i, \end{aligned}$
Step 4.	If $\ r^{i+1}\ > \varepsilon$, compute $\beta^i = \frac{\ r^{i+1}\ ^2}{\ r^i\ ^2}$, set $s^{i+1} = r^{i+1} + \beta^i s^i,$
	$i = i + 1$ and go to Step 2;
Step 5.	Set $\tilde{d}_N = p^{i+1}$ and stop.

Table 8: Truncated Newton's Algorithm

These methods require only matrix-vector products and hence they are suitable for large scale problems. The classical truncated Newton's method (TNA) (Dembo and Steihaug 1983)

was thought for the case of positive definite Hessian $\nabla^2 f(x^k)$. It is obtained by applying a CG scheme to find an approximate solution \tilde{d}_N^k to the linear system (24). We refer to *outer iterations* k for the sequence of points $\{x^k\}$ and to *inner iterations* i for the iterations of the CG scheme applied to the solution of system (24).

The inner CG scheme generates vectors d_N^i that iteratively approximate the Newton direction d_N^k . It stops either if the corresponding residual $r^i = \nabla^2 f(x^k)d_N^i + \nabla f(x^k)$ satisfies $\|r^i\| \leq \varepsilon^k$, where $\varepsilon^k > 0$ is such to enforce (31), or if a non positive curvature direction is found, that is if a conjugate direction s^i is generated such that

$$(s^i)' \nabla^2 f(x^k) s^i \leq 0. \quad (32)$$

The same convergence results of Newton's method hold, and if the smallest eigenvalue of $\nabla^2 f(x^k)$ is sufficiently bounded away from zero, the rate of convergence is still superlinear. We report in Algorithm 8 the inner CG method for finding an approximate solution of system (24). For sake of simplicity, we eliminate the dependencies on iteration k and we set $H = \nabla^2 f(x^k)$, $g = \nabla f(x^k)$. The method generates conjugate directions s^i and vectors p^i that approximate the solution \tilde{d}_N^k of the Newton's system.

By applying TNA to find an approximate solution of the linear system at Step 3 of Algorithm 5 (Algorithm 6) we obtain the monotone (nonmonotone) truncated version of the Newton's method.

In Grippo et al. (1989) a more general Truncated Newton's algorithm based on a CG method has been proposed. The algorithm attempts to find a good approximation of the Newton direction even if $\nabla^2 f(x^k)$ is not positive definite. Indeed it does not stop when a negative curvature direction is encountered and this direction is used to construct additional vectors that allow us to proceed in the CG scheme. Of course this Negative Curvature Truncated Newton's Algorithm (NCTNA) encompasses the standard TNA.

For sake of completeness, we remark that, also in the truncated setting, it is possible to define a class of line search based algorithms which are globally convergent towards points which satisfy second order NOC for Problem (2). This is done by using a curvilinear line search (28) with d^k and s^k obtained by a Lanczos based iterative scheme. These algorithms were shown to be very efficient in solving large scale unconstrained problems (Lucidi and Roma 1997, Lucidi et al. 1998b).

2.5 Quasi-Newton Methods

Quasi-Newton methods were introduced with the aim of defining efficient methods that do not require the evaluation of second order derivatives. They are obtained by setting d^k as the solution of

$$B^k d = -\nabla f(x^k), \quad (33)$$

where B^k is a $n \times n$ symmetric and positive definite matrix which is adjusted iteratively in such a way that the direction d^k tends to approximate the Newton direction. Formula (33) is referred to as the *direct* quasi-Newton formula; in turn the *inverse* quasi-Newton formula is

$$d^k = -H^k \nabla f(x^k).$$

The idea at the basis of quasi-Newton methods is that of obtaining the curvature information not from the Hessian but only from the values of the function and its gradient. The matrix B^{k+1} (H^{k+1}) is obtained as a correction of B^k (H^k), namely as $B^{k+1} = B^k + \Delta B^k$ ($H^{k+1} = H^k + \Delta H^k$). Let us denote by

$$\delta^k = x^{k+1} - x^k, \quad \gamma^k = \nabla f(x^{k+1}) - \nabla f(x^k).$$

Taking into account that, in the case of quadratic function, the so called *quasi-Newton equation*

$$\nabla^2 f(x^k)\delta^k = \gamma^k \quad (34)$$

holds, the correction ΔB^k (ΔH^k) is chosen such that

$$(B^k + \Delta B^k)\delta^k = \gamma^k, \quad (\delta^k = (H^k + \Delta H^k)\gamma^k).$$

Methods differ in the way they update the matrix B^k (H^k). Essentially they are classified according to a rank one or a rank two updating formula.

BFGS inverse quasi-Newton method

Step 1. Choose $x_0 \in \mathbb{R}^n$. Set $H^0 = I$ and $k = 0$.

Step 2. If $\nabla f(x^k) = 0$ stop.

Step 3. Set the direction

$$d^k = -H^k \nabla f(x^k);$$

Step 4. Find α^k by means of a LSA that satisfies the strong Wolfe's conditions (16) and (18).

Step 5. Set

$$\begin{aligned} x^{k+1} &= x^k + \alpha^k d^k, \\ H^{k+1} &= H^k + \Delta H^k \end{aligned}$$

with ΔH^k given by (35) with $\varphi = 1$.

Set $k = k + 1$ and go to Step 2.

Table 9: Quasi-Newton BFGS algorithm

The most used class of quasi-Newton methods is the rank two *Broyden* class. The updating rule of the matrix H^k is given by the following correction term

$$\Delta H^k = \frac{\delta^k(\delta^k)'}{(\delta^k)'\gamma^k} - \frac{H^k\gamma^k(H^k\gamma^k)'}{(\gamma^k)'H^k\gamma^k} + c(\gamma^k)'H^k\gamma^k v^k(v^k)', \quad (35)$$

where

$$v^k = \frac{\delta^k}{(\delta^k)'\gamma^k} - \frac{H^k\gamma^k}{(\gamma^k)'H^k\gamma^k},$$

and c is a nonnegative scalar. For $c = 0$ we have the *Davidon-Fletcher-Powell* (DFP) formula (Davidon 1959, Fletcher and Powell 1963) whereas for $c = 1$ we have the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) formula (Broyden 1970, Fletcher 1970b, Goldfarb 1970, Shanno 1970). An important property of methods in the Broyden class is that if H^0 is positive definite and for each k it results $(\delta^k)'\gamma^k > 0$, then positive definiteness is maintained for all the matrices H^k .

The BFGS method has been generally considered most effective. The BFGS method is a line search method, where the stepsize α^k is obtained by a LSA that enforces the strong Wolfe conditions (16) and (18). We describe a possible scheme of the inverse BFGS algorithm in Algorithm 9.

As regards convergence properties of quasi-Newton methods, satisfactory results exist for the convex case (Powell 1971, 1976, Byrd et al. 1987). In the non-convex case, only partial results exist. In particular, if there exists a constant ρ such that for every k

$$\frac{\|\gamma^k\|^2}{(\gamma^k)'\delta^k} \leq \rho, \quad (36)$$

then the sequence $\{x^k\}$ generated by the BFGS algorithm satisfies condition (13) (Powell 1976). Condition (36) holds in the convex case.

A main result on the rate of convergence of quasi-Newton methods is given in the following proposition (Dennis and Moré 1974).

Proposition 2.4 *Let $\{B^k\}$ be a sequence of non singular matrices and let $\{x^k\}$ be given by*

$$x^{k+1} = x^k - (B^k)^{-1} \nabla f(x^k).$$

Assume that $\{x^k\}$ converges to a point x^ where $\nabla^2 f(x^*)$ is non singular. Then, the sequence $\{x^k\}$ converges superlinearly to x^* if and only if*

$$\lim_{k \rightarrow \infty} \frac{\|(B^k - \nabla^2 f(x^*))(x^{k+1} - x^k)\|}{\|x^{k+1} - x^k\|} = 0.$$

For large scale problems forming and storing the matrix B^k (H^k) may be impracticable and *limited memory quasi-Newton methods* have been proposed. These methods use the information of the last few iterations for defining an approximation of the Hessian avoiding the storage of matrices. Among limited memory quasi-Newton methods, by far the most used is the Limited memory BFGS method (L-BFGS). In the L-BFGS method the matrix H^k is not formed explicitly and only pairs of vectors (δ^k, γ^k) are stored that define H^k implicitly. The product $H^k \nabla f(x^k)$ is obtained by performing a sequence of inner products involving $\nabla f(x^k)$ and the M most recent pairs (δ^k, γ^k) . Usually small values of M ($3 \leq M \leq 7$) give satisfactory results, hence the method results to be very efficient for large scale problems (Liu and Nocedal 1989, Nash and Nocedal 1991).

2.6 Derivative Free Methods

Derivative free methods neither compute nor explicitly approximate derivatives of f . They are useful when ∇f is either unavailable, or unreliable (for example, due to noise) or computationally too expensive. Nevertheless, for convergence analysis purpose, these methods assume that f is continuously differentiable.

Recently, there has been a growing interest on derivative free methods and many new results have been developed. We mention specifically only *direct search methods*, that aim to find the minimum point by using the value of the objective function on a suitable set of trial points. Direct search methods try to overcome the lack of information on the derivatives, by investigating the behaviour of the objective function in a neighborhood of the current iterate by means of samples of the objective function along a set of directions. Algorithms in this class present features which depend on the particular choice of the directions and of the sampling rule. Direct search methods are not the only derivative-free methods proposed in literature, but they include *pattern search algorithms* (PSA) (Torczon 1995, 1997, Lewis et al. 1998) and *derivative-free line search algorithms* (DFLSA) (De Leone et al. 1984, Lucidi and Sciandrone 1996) that enjoy nowadays great favor.

The algorithmic schemes for PSA and DFLSA are quite similar and differ in the assumptions on the set of directions that are used and in the rule for finding the stepsize along these directions. Both for PSA and DFLSA, we refer to a set of directions $\mathcal{D}^k = \{d^1, \dots, d^r\}$ with $r \geq n + 1$ which positively span \mathbb{R}^n . For sake of simplicity we assume that $\|d^j\| = 1$. Moreover, for PSA we assume that:

Assumption 2.5 *The directions $d^j \in \mathcal{D}^k$ are the j -th column of the matrix $(B\Gamma^k)$ with $B \in \mathbb{R}^{n \times n}$ a nonsingular matrix and $\Gamma^k \in \mathcal{M} \subseteq \mathbb{Z}^{n \times r}$ where \mathcal{M} is a finite set of full row rank integral matrices.*

PS Algorithm

Data. A set \mathcal{D}^k satisfying Assumption (2.5), $\tau \in \{1, 2\}$, $\theta = 1/2$.

Step 1. Choose $x^0 \in \mathbb{R}^n$, set $\Delta^0 > 0$ and $k = 0$;

Step 2. Check for convergence.

Step 3. If an index $j \in \{1, \dots, r\}$ exists such that

$$f(x^k + \alpha^k d^j) < f(x^k), \quad \alpha^k = \Delta^k$$

set $x^{k+1} = x^k + \alpha^k d^j$, $\Delta^{k+1} = \tau \Delta^k$, $k = k + 1$ and go to Step 2.

Step 4. Set $x^{k+1} = x^k$, $\Delta^{k+1} = \theta \Delta^k$, $k = k + 1$ and go to Step 2.

Table 10: A pattern search method

This means that in PSA the iterates x^{k+1} lie on a rational lattice “centered” in x^k . Indeed, pattern search methods are restricted in the nature of the steps s^k they take. Let \mathcal{P}^k be the set of candidates for the next iteration, namely

$$\mathcal{P}^k = \{x^{k+1} : x^k + \alpha^k d^j, \text{ with } d^j \in \mathcal{D}^k\}.$$

The set \mathcal{P}^k is called the *pattern*. The step $s^k = \alpha^k d^j$ is restricted in the values it assumes for preserving the algebraic structure of the iterates and it is chosen so as to satisfy a *simple reduction* of the objective function $f(x^k + s^k) < f(x^k)$.

As regards DFLSA we assume

Assumption 2.6 *The directions $d^j \in \mathcal{D}^k$ are the j -th column of a full row rank matrix $B^k \in \mathbb{R}^{n \times r}$.*

Hence, in DFLSA there is no underlying algebraic structure and the step s^k is a *real* vector, but it must satisfy a rule of *sufficient reduction* of the objective function.

For both schemes a possible choice of \mathcal{D}^k is $\{\pm e_j, j = 1, \dots, n\}$ where e_j is the j -th columns of the identity matrix.

We report simplified versions of PSA and DFLSA respectively in Algorithm 10 and in Algorithm 11. Both schemes produce a sequence that satisfies condition (13). We remark that by choosing at Step 3 of the PSA of Algorithm 10 the index j such that

$$f(x^k + \alpha^k d^j) = \min_{i \in \mathcal{D}^k} f(x^k + \alpha^k d^i) < f(x^k),$$

the stronger convergence result (12) can be obtained. In DFLSA condition (12) can be enforced by choosing the stepsize α^k at Step 3 of Algorithm 11 according to a derivative free LSA that enforces condition (19) together with a condition of the type

$$f\left(x^k + \frac{\alpha^k}{\delta} d^k\right) \geq \max \left\{ f(x^k + \alpha^k d^k), f(x^k) - \gamma \left(\frac{\alpha^k}{\delta}\right)^2 \right\}, \quad \delta \in (0, 1).$$

In both schemes, at Step 2 a check for convergence is needed. Of course, ∇f cannot be used for this aim. The standard test is given by

$$\left(\frac{\sum_{i=1}^{n+1} (f(x^i) - \bar{f})^2}{n+1} \right)^{1/2} \leq \textit{tolerance},$$

DFLS Algorithm

Data. A set \mathcal{D}^k satisfying Assumption (2.6), $\gamma > 0$, $\theta \in (0, 1)$.

Step 1. Choose $x^0 \in \mathbb{R}^n$, set $\Delta^0 > 0$ and $k = 0$.

Step 2. Check for convergence.

Step 3. If an index $j \in \{1, \dots, r\}$ exists such that

$$f(x^k + \alpha^k d^j) \leq f(x^k) - \gamma(\alpha^k)^2, \quad \alpha^k \geq \Delta^k$$

set $x^{k+1} = x^k + \alpha^k d^j$, $\Delta^{k+1} = \alpha^k$, $k = k + 1$ and go to Step 2.

Step 4. Set $x^{k+1} = x^k$, $\Delta^{k+1} = \theta\Delta^k$, $k = k + 1$ and go to Step 2.

Table 11: A derivative-free line search algorithm

where $\bar{f} = \frac{1}{n+1} \sum_{i=1}^{n+1} f(x^i)$, and $\{x^i, i = 1, \dots, n+1\}$ include the current point and the last n points produced along n directions.

2.7 Selected Bibliography

In the present chapter, several references have been given in correspondence to particular subjects. Here we add a list of some books and surveys of specific concern with unconstrained NLP. Of course the list is by far not exhaustive; it includes only texts widely employed and currently found.

An easy introduction to the practice of unconstrained minimization is given in McKeown et al. (1990); more technical books are the ones by Wolfe (1978) and by Kelley (1999). A basic treatise on unconstrained minimization is the book by Dennis and Schnabel (1983); to be mentioned also the survey due to Nocedal (1992). As concerns in particular conjugate gradient methods, a basic reference is Hestenes (1975). A comprehensive treatment of trust region methods is due to Conn et al. (2000). A survey paper on direct search methods is Powell (1998). A survey paper on large scale unconstrained methods is due to Nocedal (1996).

3 Constrained Optimization

We consider the basic approaches for the solution of constrained Nonlinear Programming problems. The first approach is based on the use of unconstrained minimization techniques applied to some merit function, either in sequential penalty methods or in exact penalty methods. The second approach is based on the sequential solution of quadratic programming subproblems. The third approach tries to maintain feasibility of tentative solutions, by employing feasible descent directions. Far from being exhaustive, this chapter aims to give the main ideas and tools of practical use, and to suggest how to go deeper into more technical and/or more recent results.

3.1 Introduction

We consider methods for the solution of the Nonlinear Programming (NLP) problem (1):

$$\min_{x \in \mathcal{F}} f(x),$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is an objective function, and the feasible set \mathcal{F} is described by inequality and/or equality constraints on the decision variables :

$$\mathcal{F} = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \dots, p; h_j(x) = 0, j = 1, \dots, m, m \leq n\};$$

that is, we consider methods for the solution of the *constrained* NLP problem (3):

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ & g(x) \leq 0 \\ & h(x) = 0, \end{aligned}$$

where $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$, and $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with $m \leq n$.

Here, for simplicity, the problem functions f, g, h are twice continuously differentiable in \mathbb{R}^n , even if in many cases they need to be only once continuously differentiable.

For the solution of Problem (3) two main approaches have been developed. Since, in principle, an unconstrained minimization is simpler than a constrained minimization, the first approach is based on the transformation of the constrained problem into a sequence of unconstrained problems, or even into a single unconstrained problem. Basic tools for this approach are the unconstrained minimization methods of Section 2. The second approach is based on the transformation of the constrained problem into a sequence of *simpler* constrained problems, that is either *linear* or *quadratic programming* problems; thus the basic tools become the algorithms considered in this volume for linear or quadratic programming. Due to space limits, we will confine ourselves to methods based on quadratic programming, that are by far most widely employed.

We do not treat problems with special structure, such as bound and/or linearly constrained problems, for which highly specialized algorithms exist.

We employ the additional notation $\max\{y, z\}$, where $y, z \in \mathbb{R}^p$, denotes the vector with components $\max\{y_i, z_i\}$ for $i = 1, \dots, p$.

3.2 Unconstrained Sequential Methods

In this section we consider methods that are very basic in NLP. These are: the *quadratic penalty method*, which belongs to the class of exterior penalty functions methods, the *logarithmic barrier method*, which belongs to the class of interior penalty functions or barrier functions methods, and the *augmented Lagrangian method*, which can be considered as an improved exterior penalty method.

3.2.1 The Quadratic Penalty Method

The quadratic penalty method dates back to an idea of Courant (1943), that has been fully developed in the book by Fiacco and McCormick (1968). Let us consider Problem (3) and let $p(x)$, $p : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function such that $p(x) = 0$ for all $x \in \mathcal{F}$, $p(x) > 0$ for all $x \notin \mathcal{F}$. Then, we can associate to Problem (3) the unconstrained problem:

$$\min_{x \in \mathbb{R}^n} [f(x) + \frac{1}{\epsilon} p(x)],$$

where $\epsilon > 0$. The function

$$P(x; \epsilon) = f(x) + \frac{1}{\epsilon} p(x),$$

parameterized by the *penalty parameter* ϵ , is called a penalty function for Problem (3), since it is obtained by adding to the objective function of the original problem a term that penalizes the

constraint violations. It is called an *exterior penalty function*, because its minimizers are usually exterior to the feasible set \mathcal{F} . The penalization of the constraint violations becomes more severe as the penalty parameter ϵ is smaller. Given a sequence of positive numbers $\{\epsilon^k\}$, $k = 0, 1, \dots$ such that $\epsilon^{k+1} < \epsilon^k$, $\lim_{k \rightarrow \infty} \epsilon^k = 0$, the exterior penalty method for solving Problem (3) is reported in Algorithm 12.

Exterior Penalty Function Algorithm	
Data.	$\{\epsilon^k\}$ such that $\epsilon^{k+1} < \epsilon^k$ and $\lim_{k \rightarrow \infty} \epsilon^k = 0$.
Step 1.	Choose $x^s \in \mathbb{R}^n$ and set $k = 0$.
Step 2.	Starting from x^s , find an unconstrained local minimizer
	$x^k = \arg \min_{x \in \mathbb{R}^n} P(x; \epsilon^k).$
Step 3.	If x^k is a KKT point, then stop.
Step 4.	Set $x^s = x^k$, $k = k + 1$ and go to Step 2.

Table 12: The exterior penalty function algorithm

Under weak assumptions, it can be proved that limit points of the sequence $\{x^k\}$ produced by the method of Algorithm 12 are local solutions of the constrained Problem (3).

The most widely used penalty function is the quadratic penalty function and the corresponding method is called the quadratic penalty method. The function P is obtained by using penalty terms that are the squares of the constraint violations and is given by:

$$P(x; \epsilon) = f(x) + \frac{1}{\epsilon} \{ \|\max\{g(x), 0\}\|^2 + \|h(x)\|^2 \}.$$

The quadratic penalty method is very simple; however it suffers from a main disadvantage: as ϵ becomes smaller, the Hessian matrix $\nabla^2 P(x; \epsilon)$ becomes more ill-conditioned, so that the unconstrained minimization of $P(x; \epsilon)$ becomes more difficult and slow. To alleviate the effects of ill-conditioning, the scheme of Algorithm 12 can be modified so that at Step 2 a starting point x^s closer to x^k than x^{k-1} can be given by extrapolating the trajectory of minimizers x^0, x^1, \dots, x^{k-1} . In any case, at present, the method has been overcome by methods based on the augmented Lagrangian function.

As a final remark, the function $P(x; \epsilon)$ is not twice continuously differentiable at points where some constraint g_i is active, due to the presence in P of the term $\max\{g_i, 0\}$, and this must be taken into account in the minimization of P .

3.2.2 The Logarithmic Barrier Method

The original idea of the logarithmic barrier method is due to Frisch (1955); again, in the book by Fiacco and McCormick (1968) a comprehensive treatment was given.

This method can be employed for NLP problems with only inequality constraints, that is:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ & g(x) \leq 0. \end{aligned} \tag{37}$$

It is assumed, moreover, that the *strictly feasible region*

$$\mathcal{F}^o = \{x \in \mathbb{R}^n : g_i(x) < 0, i = 1, \dots, p\} \tag{38}$$

is nonempty.

A barrier term for the feasible set \mathcal{F} of Problem (37) is a continuous function $v(x)$ defined in \mathcal{F}° , and such that $v(x) \rightarrow \infty$ as $x \in \mathcal{F}^\circ$ approaches the boundary of \mathcal{F} . Then, associated to Problem (37) we consider the unconstrained problem

$$\min_{x \in \mathcal{F}^\circ} [f(x) + \epsilon v(x)],$$

where $\epsilon > 0$. The function

$$V(x; \epsilon) = f(x) + \epsilon v(x),$$

parameterized by ϵ , is called a *barrier function* for Problem (37), since it is obtained by adding to the objective function of the problem a term that establishes a barrier on the boundary of the feasible set, thus preventing that a descent path for V starting in the interior of \mathcal{F} crosses the boundary. The barrier is as sharper as the barrier parameter ϵ is larger. Algorithms based on barrier functions are also called *interior point algorithms* because the sequences that are produced are interior to the strictly feasible region.

Given a sequence of positive numbers $\{\epsilon^k\}, k = 0, 1, \dots$ such that $\epsilon^{k+1} < \epsilon^k$, and $\lim_{k \rightarrow \infty} \epsilon^k = 0$, the barrier function method for solving Problem (37) is reported in Algorithm 13.

<p>A Barrier Function Algorithm</p> <p>Data. $\{\epsilon^k\}$ such that $\epsilon^{k+1} < \epsilon^k$ and $\lim_{k \rightarrow \infty} \epsilon^k = 0$.</p> <p>Step 1. Choose $x^s \in \mathcal{F}^\circ$ and set $k = 0$.</p> <p>Step 2. Starting from x^s, find an unconstrained local minimizer</p> $x^k = \arg \min_{x \in \mathcal{F}^\circ} V(x; \epsilon^k).$ <p>Step 3. If x^k is a KKT point, then stop.</p> <p>Step 4. Set $x^s = x^k, k = k + 1$ and go to Step 2.</p>
--

Table 13: A barrier function algorithm

As for the quadratic penalty method, the convergence analysis of the method shows that, under weak assumptions, limit points of the sequence $\{x^k\}$ produced by the method in Algorithm 13 are local solutions of the constrained Problem (37).

The most important barrier function for Problem (37) is the logarithmic barrier function, and the corresponding method is called the logarithmic barrier method. The function V is obtained by using as barrier terms the natural logarithm of the constraints:

$$V(x; \epsilon) = f(x) - \epsilon \sum_{i=1}^p \log(-g_i(x)).$$

Also the logarithmic barrier method is very simple; however the same remarks made for the quadratic penalty method hold. Namely, the method suffer of ill-conditioning as ϵ becomes smaller, and also in this case a better starting point x^s can be found by extrapolating the trajectory of minimizers x^0, x^1, \dots . An additional disadvantage is that x^s must be strictly feasible, so that the feasibility subproblem must be considered solved at the beginning. At present, the approach based on logarithmic barrier terms is not widely used for general NLP problems. However in linearly constrained optimization it is of main importance to treat bound constraints, and indeed it is at the basis of the interior points methods for linear and quadratic programming (see e.g. den Hertog (1994)).

Finally, we point out that the logarithmic barrier method can be combined with the quadratic penalty method to deal with problems with also equality constraints: in this case a penalty term $\frac{1}{\epsilon}\|h(x)\|^2$ can be added to the function V , retaining the behaviour of both approaches.

3.2.3 The Augmented Lagrangian Method

The augmented Lagrangian method has been proposed independently by Hestenes (1969) and Powell (1969) for the equality constrained problem; the extension to the inequality constrained problem is due to Rockafellar (1973). Miele et al. (1971, 1972) contributed to publicize the method; in a series of papers that are at the core of his book, Bertsekas (1982) established the superiority of this method over the quadratic penalty method.

Let us consider first a NLP problem with only equality constraints:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ & h(x) = 0. \end{aligned} \tag{39}$$

The Lagrangian function for Problem (39) is

$$L(x, \mu) = f(x) + \mu' h(x). \tag{40}$$

The *augmented Lagrangian function* for Problem (39) is the function

$$L_a(x, \mu; \epsilon) = L(x, \mu) + \frac{1}{\epsilon} \|h(x)\|^2, \tag{41}$$

obtained by adding to L the quadratic penalty term for the constraint violations.

The main property of function L_a is the following:

Proposition 3.1 *Assume that x^* is a local solution of Problem (39) that, together with the multiplier μ^* , satisfies the SOSC (restricted to equality constraints) given by Proposition 1.4 of Section 1; then there exists a threshold value ϵ^* such that, for all $\epsilon \in (0, \epsilon^*]$, x^* is a local solution of the unconstrained problem*

$$\min_{x \in \mathbb{R}^n} L_a(x, \mu^*; \epsilon). \tag{42}$$

Conversely, if for some μ^ and ϵ , x^* satisfies the SOSC to be a local solution of Problem (42) given by Proposition 1.4 of Section 1, and $h(x^*) = 0$, then x^* is a local solution of Problem (39), and μ^* is the corresponding multiplier.*

By Proposition 3.1 $L_a(x, \mu^*; \epsilon^k)$ can be employed in the quadratic penalty method of Algorithm 12 in place of $P(x; \epsilon^k)$, without requiring that ϵ^k decreases to zero. It is required only that the condition $\epsilon^k \leq \epsilon^*$ becomes fulfilled, thereby mitigating the ill-conditioning associated with the use of P . However, since μ^* is not known in advance, a procedure that iteratively updates an estimate μ^k of μ^* must be included in the method.

A generic algorithmic scheme of an augmented Lagrangian method for solving Problem (39) is in Algorithm 14.

The penalty parameter ϵ^{k+1} is usually set to $\rho\epsilon^k$, with $\rho = 1$ if $\|h(x^k)\|^2$ is sufficiently smaller than $\|h(x^{k-1})\|^2$, and $\rho < 1$ otherwise.

Updating formulas for the multiplier estimate are based on the fact that, under the assumptions of Proposition 3.1, the pair (x^*, μ^*) is a saddle point of the function L_a , so that an ascent strategy with respect to the *dual function*

$$D_a(\mu; \epsilon) = \min_{x \in \mathcal{B}_{x^*}} L_a(x, \mu; \epsilon),$$

An Augmented Lagrangian Algorithm

Step 1. Choose ϵ^0 , μ^0 and $x^s \in \mathbb{R}^n$; set $k = 0$.

Step 2. Starting from x^s , find an unconstrained local minimizer

$$x^k = \arg \min_{x \in \mathbb{R}^n} L_a(x, \mu^k; \epsilon^k)$$

Step 3. If (x^k, μ^k) is a KKT pair, then stop.

Step 4. Choose a penalty parameter $\epsilon^{k+1} \in (0, \epsilon^k]$.

Step 5. Update the multiplier estimate to μ^{k+1} .

Step 6. Set $x^s = x^k$, $k = k + 1$ and go to Step 2.

Table 14: An Augmented Lagrangian algorithm

is effective. The simplest updating formula is the first order formula $\mu^{k+1} = \mu^k + \frac{2}{\epsilon^k} h(x^k)$, which corresponds to a gradient ascent step for D_a ; second order formulas are based on a Newton's step.

By the augmented Lagrangian method of Algorithm 14 we get a sequence $\{x^k\}$ and a sequence $\{\mu^k\}$. If the sequence $\{\mu^k\}$ is bounded, the sequence $\{x^k\}$ has the same convergence properties of the sequence produced by the quadratic penalty method of Algorithm 12. The rate of convergence depends on the updating formula employed for the multiplier.

Let us consider now a NLP problem with only inequality constraints, that is Problem (37). An inequality constraint $g_i(x) \leq 0$ can be transformed into the equality constraint $g_i(x) + y_i^2 = 0$ by using an additional squared slack variable y_i^2 . Problem (37) can be rewritten as the equality constrained problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^p} \quad & f(x) \\ & g_i(x) + y_i^2 = 0, \quad i = 1, \dots, p. \end{aligned} \quad (43)$$

This transformation is only instrumental; indeed, for algorithmic purposes, it has the disadvantage of increasing the number of variables from n to $n + p$.

Let λ_i be the multiplier associated with the i -th constraint; then the augmented Lagrangian function for the equality constrained Problem (43) turns out to be:

$$L_a(x, y, \lambda; \epsilon) = f(x) + \sum_{i=1}^p \lambda_i (g_i(x) + y_i^2) + \frac{1}{\epsilon} \sum_{i=1}^p (g_i(x) + y_i^2)^2. \quad (44)$$

The function $L_a(x, y, \lambda; \epsilon)$ could be employed in the algorithmic scheme described in Algorithm 14, by minimizing with respect to x, y for given λ and ϵ . However, the structure of L_a in (44) reveals that the minimization with respect to each y_i^2 can be easily performed in closed form. Given x, λ and ϵ , the minimizer y_i^2 of L_a is given by $y_i^2(x, \lambda; \epsilon) = -\min \left\{ 0, g_i(x) + \frac{\epsilon}{2} \lambda_i \right\}$. By substituting in (44) we get the augmented Lagrangian function in the original variables of Problem (37):

$$L_a(x, \lambda; \epsilon) = f(x) + \lambda' \max \left\{ g(x), -\frac{\epsilon}{2} \lambda \right\} + \frac{1}{\epsilon} \left\| \max \left\{ g(x), -\frac{\epsilon}{2} \lambda \right\} \right\|^2. \quad (45)$$

Accordingly, the first order formula for updating the multiplier λ is $\lambda^{k+1} = \max \left\{ 0, \lambda^k + \frac{2}{\epsilon^k} g(x^k) \right\}$. Note that function (45) is not twice continuously differentiable at

points where $g_i(x) = -\frac{\epsilon}{2}\lambda_i$ for some i ; however the lack of second order differentiability does not occur locally in a neighborhood of a KKT pair (x^*, λ^*) if $g(x^*), \lambda^*$ satisfy the strict complementarity condition.

Problems with both equality and inequality constraints can be handled by combining the two cases.

The augmented Lagrangian method with the first order multiplier updating is also referred to as the *multiplier method*. A less used term is the *shifted barrier method*.

Vast computational experience shows that the augmented Lagrangian function approach is reasonably effective, and many commercial softwares for the solution of NLP problems are based on it.

3.3 Unconstrained Exact Penalty Methods

Exact penalty methods attempt to solve NLP problems by means of a single minimization of an unconstrained function, rather than by means of a sequence of unconstrained minimizations. Exact penalty methods are based on the construction of a function depending on a penalty parameter $\epsilon > 0$, such that the unconstrained minimum points of this function are also solutions of the constrained problem for all values of ϵ in the interval $(0, \epsilon^*]$, where ϵ^* is a threshold value.

We can subdivide exact penalty methods into two classes: methods based on *exact penalty functions* and methods based on *exact augmented Lagrangian functions*. The term exact penalty function is used when the variables of the unconstrained problem are the same of the original constrained problem, whereas the term exact augmented Lagrangian function is used when the unconstrained problem is defined in the product space of the problem variables and of the multipliers (primal and dual variables).

An exact penalty function, or an augmented Lagrangian function, possesses different exactness properties, depending on which kind of correspondence can be established, under suitable assumptions, between the set of the local (global) solutions of the constrained problem and the set of local (global) minimizers of the unconstrained function. Different definitions of exactness can be found in Di Pillo and Grippo (1989).

The simplest function that possesses exactness properties is obtained by adding to the objective function the ℓ_1 norm of the constraint violations, weighted by a penalty parameter ϵ . The so-called ℓ_1 *exact penalty function* for Problem (3), introduced by Zangwill (1967), is given by:

$$L_1(x; \epsilon) = f(x) + \frac{1}{\epsilon} \left\{ \sum_{i=0}^p \max\{g_i(x), 0\} + \sum_{j=1}^m |h_j(x)| \right\}. \quad (46)$$

In spite of its simplicity, function (46) is not continuously differentiable, so that its unconstrained minimization cannot be performed using the techniques seen in Section 2 of this volume; one should resort to the less efficient *nonsmooth optimization* techniques.

Continuously differentiable exact penalty functions can be obtained from the augmented Lagrangian function of §3.2.3 by substituting the multiplier vectors λ, μ by continuously differentiable *multiplier functions* $\lambda(x), \mu(x)$, with the property that $\lambda(x^*) = \lambda^*, \mu(x^*) = \mu^*$ whenever the triplet (x^*, λ^*, μ^*) satisfies the KKT necessary conditions.

For the equality constrained problem (39) a multiplier function is

$$\mu(x) = -[\nabla h(x)' \nabla h(x)]^{-1} \nabla h(x)' \nabla f(x);$$

then *Fletcher's exact penalty function*, introduced in Fletcher (1970a), is:

$$U_F(x; \epsilon) = f(x) + \mu(x)' h(x) + \frac{1}{\epsilon} \|h(x)\|^2. \quad (47)$$

Similarly, for the inequality constrained problem (37) a multiplier function is:

$$\lambda(x) = -[\nabla g(x)' \nabla g(x) + \theta G^2(x)]^{-1} \nabla g(x)' \nabla f(x),$$

where $\theta > 0$ and $G(x)$ is the diagonal matrix with entries $g_i(x)$; an exact penalty function is (Glad and Polak 1979, Di Pillo and Grippo 1985):

$$U(x; \epsilon) = f(x) + \lambda'(x) \max \left\{ g(x), -\frac{\epsilon}{2} \lambda(x) \right\} + \left\| \max \left\{ g(x), -\frac{\epsilon}{2} \lambda(x) \right\} \right\|^2. \quad (48)$$

Exact augmented Lagrangian functions are obtained by incorporating in the augmented Lagrangian function L_a terms that penalize the violation of necessary optimality conditions different than feasibility. For instance, for the equality constrained problem (39) an exact augmented Lagrangian function, studied by Di Pillo and Grippo (1979) is:

$$S_a(x, \mu; \epsilon) = f(x) + \mu' h(x) + \frac{1}{\epsilon} \|h(x)\|^2 + \theta \|\nabla h(x)' \nabla_x L(x, \mu)\|^2, \quad (49)$$

where $\theta > 0$. It is evident that the last term in the r.h.s. of (49) penalizes in some way the violation of the necessary optimality condition $\nabla_x L(x, \mu) = 0$.

Exact penalty functions and exact augmented Lagrangian functions can also be endowed with barrier terms that act on the boundary of a set that strictly contains the feasible set. In this way it is possible to build functions that not only have as unconstrained minimizers the solution of the constrained problem, but have also compact level sets. Both these properties are quite important for the design of globally convergent algorithms. Indeed, making use of these functions it is possible to develop algorithms for constrained problems that behave like the algorithms for unconstrained problems in terms of convergence and convergence rate, see for instance Di Pillo et al. (1980, 1982, 1986).

Algorithms based on continuously differentiable exact penalty functions have not been widely implemented. This is mainly due to the fact that the evaluation of the multiplier function requires the inversion of a matrix of the dimension of the constraints vector. More promising is the use of exact augmented Lagrangian functions, even if their minimization must be performed on the enlarged space of the decision variables and of the multipliers.

Functions that possess exactness properties are also referred to as *exact merit functions* when they are used in sequential algorithms to measure the merit of subsequent iterates in moving towards the solution of the constrained problem; in fact their minimization combines the often conflicting requirements of reducing both the objective function and the constraint violations (and the violation of other NOC).

Exact penalty methods require in any case some strategy to select a penalty parameter value in the interval $(0, \epsilon^*]$ where exactness holds.

3.4 Sequential Quadratic Programming Methods

The *sequential quadratic programming* (SQP) approach can be viewed as a generalization to constrained optimization of Newton's method for unconstrained optimization: in fact it finds a step away from the current point by minimizing a quadratic model of the problem. The SQP approach has been introduced by Wilson (1963); main developments are due to Garcia-Palomares and Mangasarian (1976), Han (1976, 1977), Powell (1978a,c,b).

For simplicity, let us consider the equality constrained Problem (39), with Lagrangian function (40). In principle, a solution of this problem could be found by solving w.r.t. (x, μ) , the KKT first order NOC:

$$\nabla L(x, \mu) = \begin{pmatrix} \nabla f(x) + \nabla h(x) \mu \\ h(x) \end{pmatrix} = 0. \quad (50)$$

The Newton iteration for the solution of (50) is given by

$$\begin{pmatrix} x^{k+1} \\ \mu^{k+1} \end{pmatrix} = \begin{pmatrix} x^k \\ \mu^k \end{pmatrix} + \begin{pmatrix} d_x^k \\ d_\mu^k \end{pmatrix},$$

where the Newton's step (d_x^k, d_μ^k) solves the so called *KKT system*:

$$\begin{pmatrix} \nabla_x^2 L(x^k, \mu^k) & \nabla h(x^k) \\ \nabla h(x^k)' & 0 \end{pmatrix} \begin{pmatrix} d_x \\ d_\mu \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) + \nabla h(x^k)\mu^k \\ h(x^k) \end{pmatrix}.$$

By adding the term $\nabla h(x^k)\mu^k$ to both sides of the first equation of the KKT system, the Newton's iteration for (50) is determined equivalently by means of the equations:

$$x^{k+1} = x^k + d_x^k, \quad (51)$$

$$\begin{pmatrix} \nabla_x^2 L(x^k, \mu^k) & \nabla h(x^k) \\ \nabla h(x^k)' & 0 \end{pmatrix} \begin{pmatrix} d_x^k \\ \mu^{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ h(x^k) \end{pmatrix}. \quad (52)$$

The iteration defined by (51–52) constitute the so called *Newton-Lagrange method* for solving Problem (39).

Consider now the quadratic programming (QP) problem:

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \frac{1}{2} s' \nabla_x^2 L(x^k, \mu^k) s + \nabla f(x^k)' s \\ & \nabla h(x^k)' s + h(x^k) = 0. \end{aligned} \quad (53)$$

The first order NOC for (53) can be written as:

$$\begin{pmatrix} \nabla_x^2 L(x^k, \mu^k) & \nabla h(x^k) \\ \nabla h(x^k)' & 0 \end{pmatrix} \begin{pmatrix} s \\ \eta \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ h(x^k) \end{pmatrix}, \quad (54)$$

where η is a multiplier vector for the linear equality constraints of Problem (53). By comparing (52) with (54) we see that (d_x^k, μ^{k+1}) and a solution (s^k, η^k) of (54) satisfy the same system. Therefore, in order to solve Problem (39) we can employ an iteration based on the solution of the QP problem (53) rather than on the solution of the linear system (52).

The SQP method for Problem (39) in its simplest form is described in Algorithm 15.

SPQ method: equality constraints

- Step 1.** Choose an initial pair (x^0, μ^0) ; set $k = 0$.
- Step 2.** If (x^k, μ^k) is a KKT pair of Problem (39), then stop.
- Step 3.** Find (s^k, η^k) as a KKT pair of Problem (53).
- Step 4.** Set $x^{k+1} = x^k + s^k$; $\mu^{k+1} = \eta^k$, $k = k + 1$ and go to Step 2.

Table 15: A Sequential Quadratic Programming method for equality constrained problems

The SQP method is preferable to the Newton-Lagrange method for the following reason. Assume that the sequence $\{(x^k, \mu^k)\}$ produced by the Newton-Lagrange method converges to some (x^*, μ^*) ; then, in the limit, $d_x = 0$, and from (52) it results, as expected, that (x^*, μ^*) satisfies the first order NOC for Problem (39). Assume now that the sequence $\{(x^k, \mu^k)\}$ produced by the SQP method converges to some (x^*, μ^*) ; then, in the limit, $s = 0$, and from (54) it results again that (x^*, μ^*) satisfies the first order NOC; however from (53) and (54)

we have also, in the limit, that $s'\nabla^2 L_x(x^*, \mu^*)s \geq 0$ for all $s \in \mathbb{R}^n$ such that $\nabla h(x^*)'s = 0$; therefore, in this case, (x^*, μ^*) satisfies also a second order NOC. In essence, the minimization process of SQP drives the sequence toward beneficial KKT solutions.

Note that, thanks to the constraint, the objective function of the quadratic problem (53) could also be written as:

$$\frac{1}{2}s'\nabla_x^2 L(x^k, \mu^k)s + \nabla_x L(x^k, \mu^k)'s + L(x^k, \mu^k).$$

This leads to an alternative motivation of the SQP method. Indeed, Problem (39) can be written equivalently as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & L(x, \mu^*) \\ & h(x) = 0, \end{aligned}$$

where μ^* is the KKT multiplier associated to the solution x^* of (39). Then solving Problem (53) corresponds to finding (x^*, μ^*) by iteratively solving the problem of minimizing a quadratic approximation of the Lagrangian $L(x, \mu^k)$ subject to a linear approximation of the constraint $h(x) = 0$.

The last observation suggests that the SQP framework can be extended easily to the general NLP problem (3). Indeed, by linearizing also the inequality constraints we get the quadratic programming problem:

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \frac{1}{2}s'\nabla_x^2 L(x^k, \lambda^k, \mu^k)s + \nabla f(x^k)'s \\ & \nabla g(x^k)'s + g(x^k) \leq 0 \\ & \nabla h(x^k)'s + h(x^k) = 0. \end{aligned} \tag{55}$$

The inequality constrained case puts in evidence an additional advantage of the SQP approach over the Newton-Lagrange method: indeed it is much simpler to solve the QP subproblem (55) than to solve the set of equalities and inequalities obtained by linearizing the KKT conditions for Problem (3).

The SQP algorithm for Problem (3) is given in Algorithm 16.

SPQ method
Step 1. Choose an initial triplet (x^0, λ^0, μ^0) ; set $k = 0$.
Step 2. If (x^k, λ^k, μ^k) is a KKT triplet of Problem (3), then stop.
Step 3. Find (s^k, ζ^k, η^k) as a KKT triplet of Problem (55).
Step 4. Set $x^{k+1} = x^k + s^k$; $\lambda^{k+1} = \zeta^k$; $\mu^{k+1} = \eta^k$, $k = k + 1$ and go to Step 2.

Table 16: A Sequential Quadratic Programming method for general constrained problems

The equivalence between the SQP method and the Newton-Lagrange method provides the basis for the main convergence and rate of converge results of the former.

Proposition 3.2 *Assume that f, g, h are twice differentiable with Lipschitz continuous second derivatives; assume that (x^*, λ^*, μ^*) satisfy the SOSC of Proposition 1.4 of Section 1 under strict complementarity between $g(x^*)$ and λ^* . Then, there exists an open neighborhood $\mathcal{B}_{(x^*, \lambda^*, \mu^*)}$ such that, if $(x^0, \lambda^0, \mu^0) \in \mathcal{B}_{(x^*, \lambda^*, \mu^*)}$, the sequence $\{(x^k, \lambda^k, \mu^k)\}$ produced by the SQP method is well defined and converges to (x^*, λ^*, μ^*) with quadratic rate.*

The SQP method outlined above requires the computation of the second order derivatives that appear in $\nabla_x^2 L(x^k, \lambda^k, \mu^k)$. As in the Newton's method in unconstrained optimization, this can be avoided by substituting the Hessian matrix $\nabla_x^2 L(x^k, \lambda^k, \mu^k)$ with a quasi-Newton approximation B^k , which is updated at each iteration. An obvious update strategy would be to define:

$$\delta^k = x^{k+1} - x^k, \quad \gamma^k = \nabla L(x^{k+1}, \lambda^k, \mu^k) - \nabla L(x^k, \lambda^k, \mu^k),$$

and to update the matrix B^k by using the BFGS formula

$$B^{k+1} = B^k + \frac{\gamma^k (\gamma^k)'}{(\gamma^k)' \delta^k} - \frac{B^k \delta^k (\delta^k)' B^k}{(\delta^k)' B^k \delta^k} \quad (56)$$

for unconstrained optimization. However, one of the properties that make the BFGS formula appealing for unconstrained problems, that is maintenance of positive definiteness of B^k , is no longer assured, since $\nabla_x^2 L(x^*, \lambda^*, \mu^*)$ is usually positive definite only on a subset of \mathbb{R}^n . This difficulty may be overcome by modifying γ^k as proposed by Powell (1978b). Let:

$$\tilde{\gamma}^k = \tau \gamma^k + (1 - \tau) B^k \delta^k,$$

where the scalar τ is defined as

$$\tau = \begin{cases} 1 & \text{if } (\delta^k)' \gamma^k \geq 0.2 (\delta^k)' B^k \delta^k \\ 0.8 \frac{(\delta^k)' B^k \delta^k}{(\delta^k)' B^k \delta^k - (\delta^k)' \gamma^k} & \text{if } (\delta^k)' \gamma^k < 0.2 (\delta^k)' B^k \delta^k, \end{cases}$$

and update B^k by the *damped* BFGS formula, obtained by substituting $\tilde{\gamma}^k$ for γ^k in (56); then, if B^k is positive definite the same holds for B^{k+1} . This ensures also that, if the quadratic programming problem is feasible, it admits a solution. Of course, by employing the quasi-Newton approximation, the rate of convergence becomes superlinear.

Proposition 3.2 states a local convergence result. If, as often happens, a point $(x^0, \lambda^0, \mu^0) \in \mathcal{B}_{(x^*, \lambda^*, \mu^*)}$ is not known, the following difficulties have to be tackled:

- at iteration k the QP subproblem may have no solution;
- the sequence $\{(x^k, \lambda^k, \mu^k)\}$ may not be convergent.

The first difficulty arises either because the quadratic objective function is *unbounded from below* on the linearized feasible set, or because the linearized feasible set *is empty*. The first occurrence can be avoided by substituting $\nabla_x^2 L(x^k, \lambda^k, \mu^k)$ with a positive definite approximation B^k , for instance as described before; or by employing the Hessian of the augmented Lagrangian $\nabla_x^2 L_a(x^k, \lambda^k, \mu^k; \epsilon)$ in place of $\nabla_x^2 L(x^k, \lambda^k, \mu^k)$. The second occurrence can be overcome by defining a suitable relaxation of the QP subproblem so as to obtain a problem which is always feasible, even if of larger dimension. As an example, the following subproblem has been proposed (De O. Pantoja and Mayne 1991):

$$\begin{aligned} \min_{s \in \mathbb{R}^n, w \in \mathbb{R}} \quad & \frac{1}{2} s' B^k s + \nabla f(x^k)' s + \frac{1}{\epsilon} w \\ & \nabla g_i(x^k)' s + g_i(x^k) \leq w, \quad i = 1, \dots, p \\ & |\nabla h_j(x^k)' s + h_j(x^k)| \leq w, \quad j = 1, \dots, m \\ & w \geq 0, \end{aligned} \quad (57)$$

where $\epsilon > 0$ is a suitable penalty parameter. It is easy to verify that the feasible set of Problem (57) is not empty, so that, if B^k is positive definite, the problem admits a solution.

With respect to the convergence of the sequence $\{(x^k, \lambda^k, \mu^k)\}$, we observe preliminarily that if $\{x^k\}$ converges to some x^* where the linear independence constraints qualification (LICQ) holds, then $\{(\lambda^k, \mu^k)\}$ also converges to some (λ^*, μ^*) such that (x^*, λ^*, μ^*) satisfies the KKT

NOC for Problem (3). In order to enforce the convergence of the sequence $\{x^k\}$, in the same line of the Newton's method for unconstrained optimization, two strategies have been devised with reference to a suitable *merit function* associated with the NLP problem: the *line search* approach and the *trust region* approach. As said before, a merit function aims to reduce both the objective function and the constraint violations (as well as the violation of other necessary optimality conditions); in this way it provides a measure of the progress toward a solution of the constrained problem.

3.4.1 The Line Search Approach

In the line search approach the sequence $\{x^k\}$ is given by the iteration:

$$x^{k+1} = x^k + \alpha^k s^k,$$

where the direction s^k is obtained by solving the k -th QP subproblem, and the stepsize α^k is evaluated so as to get a sufficient decrease of the merit function. A merit function widely employed in SQP methods is the ℓ_1 exact penalty function $L_1(x; \epsilon)$ given by (46). Although L_1 is not differentiable everywhere, it always has a directional derivative. It can be shown that, if the quadratic form in the QP subproblem is positive definite, and if the penalty parameter ϵ in L_1 is sufficiently small, then s^k is a descent direction at x^k for L_1 , so that an Armijo-like line search along s^k can be performed (Han 1977). However, due to its non differentiability, the merit function L_1 suffers from a main drawback known as the *Maratos effect* (Maratos 1978): even close to a solution, the stepsize $\alpha^k = 1$ may not be accepted, thus deteriorating the convergence rate of the pure SQP method. To avoid the Maratos effect, different techniques have been proposed. Among them a *curvilinear line search* (Mayne and Polak 1982), based on the formula $x^{k+1} = x^k + \alpha^k s^k + (\alpha^k)^2 \tilde{s}^k$, where the direction \tilde{s}^k , that provides a *second order correction*, is evaluated so as to further reduce the constraint violations; and the *watchdog technique* (Chamberlain et al. 1982), based on nonmonotone strategies, by which we allow the merit function L_1 to increase on certain iterations.

The Maratos effect can also be overcome by employing merit functions that are continuously differentiable. For equality constraints, the use of Fletcher's exact penalty function U_F given by (47) is usually referred (Powell and Yuan 1986); the extension to inequality constraints is based on the exact penalty function U given in (48) (Di Pillo et al. 1992, Facchinei 1997). It can be shown that for a sufficiently small value of the penalty parameter ϵ , s^k gives a descent direction for the continuously differentiable exact penalty functions at x^k , and eventually the stepsize $\alpha = 1$ is accepted, so that it is possible to conciliate global convergence and superlinear convergence rate. Continuously differentiable exact augmented Lagrangian functions have also been studied as merit functions in SQP methods, with similar results (Dixon 1979, Lucidi 1990). For instance, when ϵ is small, the direction given by $(s^k, (\eta^k - \mu^k))$ is a descent direction at (x^k, μ^k) for the function $S_a(x, \mu; \epsilon)$ given by (49). Continuously differentiable exact merit functions are very attractive for their theoretical properties, but have not been widely used yet, mainly because of the fact that the computations needed are rather cumbersome.

We mention that also the simple augmented Lagrangian function has been used as a merit function; however, as it appears for instance from Proposition 3.1, the augmented Lagrangian function is an exact merit function only if proper values of the multipliers are selected. This fact makes arguments on global convergence questionable.

3.4.2 The Trust Region Approach

As in the unconstrained case, the trust region approach consists in introducing in the quadratic subproblem a trust region constraint that bounds the region where the subproblem is trusted

to represent well the original constrained problem:

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \frac{1}{2} s' \nabla_x^2 L(x^k, \lambda^k, \mu^k) s + \nabla f(x^k)' s \\ & \nabla g(x^k)' s + g(x^k) \leq 0 \\ & \nabla h(x^k)' s + h(x^k) = 0 \\ & \|s\|^2 \leq (a^k)^2. \end{aligned} \tag{58}$$

However, due to the presence of the linear constraints, the solution of the trust region subproblem (58) is much more difficult than in the unconstrained case, and may even not exist at all. A possibility, proposed by Fletcher (1981), is to take into account the linear constraints by ℓ_1 penalty terms, and define the trust region by means of the ℓ_∞ norm of s ; thus the problem becomes:

$$\begin{aligned} \min_{s \in \mathbb{R}^n} \quad & \frac{1}{2} s' \nabla_x^2 L(x^k, \lambda^k, \mu^k) s + \nabla f(x^k)' s \\ & + \frac{1}{\epsilon^k} \sum_{i=1}^p \max \{ \nabla g_i(x^k)' s + g_i(x^k), 0 \} \\ & + \frac{1}{\epsilon^k} \sum_{j=1}^m |\nabla h_j(x^k)' s + h_j(x^k)| \\ & -a^k \leq s \leq a^k. \end{aligned} \tag{59}$$

It is easily seen that Problem (59) admits always a solution s^k , even when $\nabla_x^2 L(x^k, \lambda^k, \mu^k)$ is substituted by some approximation to avoid the computation of second order derivatives. Moreover, by suitable manipulations, Problem (59) can be put in the form of a QP problem that can be solved by standard algorithms.

The $L_1(x; \epsilon)$ exact penalty function is used as a merit function. The objective function of Problem (59) is used as an approximation of the L_1 function in the trust region. Similarly to the unconstrained case, the step s^k is accepted if the ratio between the actual and the predicted reduction in the L_1 function is not too small; otherwise the trust region radius a^k is decreased, and the problem is re-solved to obtain a new candidate step. Once $x^{k+1} = x^k + s^k$ has been determined, some multiplier updating formula is employed to get the new multiplier estimates $(\lambda^{k+1}, \mu^{k+1})$. Moreover some updating rule for the penalty parameter is needed to define the next subproblem in (59). The resulting algorithm is known as the SL_1 QP algorithm. The SL_1 QP algorithm has good global convergence properties; however it also suffers of the Maratos effect, that can be overcome by techniques similar to the ones described for the L_1 merit function in the line search approach.

Finally, we mention the *feasible sequential quadratic programming* algorithms, which force all iterates to be feasible. Feasibility may be required when the objective function f is not defined outside the feasible set, or when termination at an infeasible point (which may happen in practice with most algorithms) is undesirable. To maintain feasibility, while retaining fast local convergence properties, the step is obtained along a direction that combines the QP direction, a direction that points into the interior of the feasible set, and possibly a second-order correction direction (Paniers and Tits 1993). Of course feasible SQP algorithms are more computationally expensive than the standard SQP algorithms, but they have the additional advantage that the objective function f itself can be used as a merit function.

3.5 Feasible Direction Methods

For NLP problems with only linear constraints several methods can be devised based on search directions d^k that are both *of descent* for the objective function and *feasible* for the constraints, that is directions such that, if $x^k \in \mathcal{F}$ is not a minimizer, then not only $f(x^k + \alpha d^k) < f(x^k)$, but also $x^k + \alpha d^k \in \mathcal{F}$ for small enough values of the stepsize α . For instance, the *feasible*

direction method of Zoutendijk (1960), the *gradient projection method* of Rosen (1960, 1961), the *reduced gradient method* of Wolfe (1963), and the more recent *interior points methods*, see e.g. Ye (1997).

The topic of linearly constrained NLP problem is highly specialized and strictly related to quadratic programming, which just represents a particular case. Even a short review would exceed the space allowed for this chapter. For a comprehensive treatment we may refer to Chapter 10 of Bazaraa et al. (1993).

Here we mention that many attempts have been made to generalize feasible direction methods to problems with general constraints, usually by linearizing the constraints in the neighborhood of the current point x^k . However, in doing so, the difficulty arises that, if x^{k+1} is feasible for the linearized constraints, it may not be feasible for the original constraints, specially when nonlinear equality constraints are present. In this case the direction d^k may be tangent to some constraints and an additional correction step must be evaluated so as to restore feasibility while retaining a decrease in the objective function; how to perform this *restoration step* is the critical issue in feasible direction methods for general NLP problems.

We shall shortly describe only the *generalized reduced gradient method* (Abadie and Carpentier 1969), since it is implemented in some currently available codes for NLP.

In this context, it is usual to consider a NLP problem in the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ & h(x) = 0 \\ & l \leq x, \end{aligned} \tag{60}$$

where some components of the n -vector l can be $-\infty$. Problem (3) can be put in the form (60) by introducing linear slack variables y_i , so that the inequality constraint $g_i(x) \leq 0$ is substituted by the equality constraint $g_i(x) + y_i = 0$ plus the bound constraint $y_i \geq 0$.

Assume that x^k is a given feasible point. Under the LICQ assumption and after some reordering, we can partition x^k as $[(x_B^k)' | (x_N^k)']'$, where $x_B^k \in \mathbb{R}^m$ is a subset of *basic* variables with the property that the gradient matrix $\nabla_{x_B} h(x^k)$ is non singular, and $x_N^k \in \mathbb{R}^{n-m}$ is a subset of *non basic* variables. Accordingly l is partitioned into $[(l_B)' | (l_N)']'$ and for simplicity, the *nondegeneracy assumption* that $l_B < x_B^k$ is made. By the implicit function theorem, there exists a neighborhood $\mathcal{B}_{x_N^k} \subseteq \mathbb{R}^{n-m}$ of x_N^k and a continuously differentiable function $z^k : \mathbb{R}^{n-m} \rightarrow \mathbb{R}^m$ such that $x_B^k = z^k(x_N^k)$ and $h(z^k(x_N), x_N) = 0$ for all $x_N \in \mathcal{B}_{x_N^k}$. Then, locally in a neighborhood of x^k , Problem (60) can be rewritten as

$$\begin{aligned} \min_{x_N \in \mathbb{R}^{n-m}} \quad & f(z^k(x_N), x_N) \\ & l_N \leq x_N \\ & l_B \leq z^k(x_N). \end{aligned} \tag{61}$$

By the nondegeneracy assumption, the last bounds are non active at x_N^k . Hence a search direction $d_N^k \in \mathbb{R}^{n-m}$ at x_N^k can be determined by using algorithms for simple bound constrained problems, making use of the *reduced gradient* formula:

$$\nabla_{x_N} f(z^k(x_N), x_N) = \nabla_{x_N} f(x) - (\nabla_{x_N} h(x) \nabla_{x_B} h(x)^{-1}) \nabla_{x_B} f(x).$$

Algorithms for bound constrained problems are highly specialized and can be obtained by adapting the unconstrained methods of Section 2.

Given x_N^k and d_N^k , a line search procedure that uses the objective function f as a merit function is adopted. Letting $x_N(\alpha) = x_N^k + \alpha d_N^k$, in the line search f needs to be evaluated at $(z^k(x_N(\alpha)), x_N(\alpha))$. To this aim the value of $z^k(x_N(\alpha))$ is obtained by applying Newton's method to solve w.r.t. x_B the constraint $h(x_B, x_N) = 0$; in doing this, care must also be taken

that the bound constraint on x_B is not violated. In this way, feasibility of the successive iterates is maintained.

It is easily understood that the convergence arguments of feasible direction methods for general NLP problems are quite involved.

Finally, we point out that feasible direction methods require that a feasible starting point is available, and this is not always easy for general nonlinear constraints; on the other hand, since the feasibility subproblem is considered solved, these methods do not rely on merit functions depending on penalty parameters.

3.6 Selected Bibliography

In the present chapter, several references have been given in correspondence to particular subjects. Here we add a list of some books and surveys of specific concern with constrained NLP.

The sequential penalty approach, both exterior and interior, is fully developed in the classical book by Fiacco and McCormick (1968) that has been recently revisited by Nash (1998). For interior point methods related to the logarithmic barrier approach, and specialized for linear and quadratic programming problems, we may refer to den Hertog (1994), Roos et al. (1997), Ye (1997), Wright (1997), and the recent survey by Freund and Mizuno (2000); a particular emphasis on theoretical aspects is given in Nesterov and Nemirovskii (1994). For the augmented Lagrangian method the basic reference is the book by Bertsekas (1982). Exact penalty methods have been surveyed by Di Pillo (1994). A survey on penalty, exact penalty and augmented Lagrangian methods is due to Boukary and Fiacco (1995). The sequential quadratic programming has been surveyed by Boggs and Tolle (1995), and is extensively treated in Conn et al. (2000). A particular attention to feasible direction methods is given in the book by Bazararaa et al. (1993). Algorithms for large scale problems are surveyed in Conn et al. (1994).

References

- J. Abadie and J. Carpentier. Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints. In R. E. Fletcher, editor, *Optimization*, pages 37–47. Academic Press, 1969.
- M. Al-Baali. Descent properties and global convergence of the Fletcher-Reeves method with inexact line search. *IMA Journal on Numerical Analysis*, 5:121–124, 1985.
- M. Avriel. *Nonlinear Programming Analysis and Methods*. Prentice-Hall, 1976.
- J. Barzilai and J. M. Borwein. Two point step size gradient method. *IMA Journal on Numerical Analysis*, 8:141–148, 1988.
- M. S. Bazararaa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming - Theory and Algorithms (2nd edition)*. John Wiley & Sons, 1993.
- D. P. Bertsekas. *Nonlinear Programming (2nd edition)*. Athena Scientific, 1999.
- D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.
- P. T. Boggs and J. W. Tolle. Sequential Quadratic Programming. *Acta Numerica*, pages 1–51, 1995.
- J. F. Bonnans, J. C. Gilbert, C. Lemarèchal, and C. Sagastizàbal. *Optimisation Numérique: Aspects théoriques et pratiques*. Springer Verlag, 1997.

- D. Boukary and A.V. Fiacco. Survey of penalty, exact-penalty and multiplier methods from 1968 to 1993. *Optimization*, 32:301–334, 1995.
- C. G. Broyden. The convergence of a class of double-rank minimization algorithms, part I and II. *Journal of the Institute of Mathematics and its Applications*, 6:76–90 and 222–231, 1970.
- R. H. Byrd, J. Nocedal, and Y. Yuan. Global convergence of a class of quasi-Newton methods on convex problems. *SIAM Journal on Numerical Analysis*, 24:1171–1190, 1987.
- R. Chamberlain, C. Lemarechal, H. C. Pedersen, and M. J. D. Powell. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Mathematical Programming Study*, 16:1–17, 1982.
- A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust Region Methods*. SIAM, 2000.
- A.R. Conn, N. Gould, and P.L. Toint. Large-scale nonlinear constrained optimization: a current survey. In E. Spedicato, editor, *Algorithms for Continuous Optimization: the State of the Art*, pages 287–332. Kluwer Academic Publishers, 1994.
- R. Courant. Variational methods for the solution of problems with equilibrium and vibration. *Bull. Amer. Math. Soc.*, 49:1–23, 1943.
- W. C. Davidon. Variable metric method for minimization. Technical Report ANL 5990 (revised), Argonne National Laboratory, 1959. Published in *SIAM Journal On Optimization*, 1, 1–17,(1991).
- R. De Leone, M. Gaudioso, and L. Grippo. Stopping criteria for linesearch methods without derivatives. *Mathematical Programming*, 30:285–300, 1984.
- J. F. A. De O. Pantoja and D. Q. Mayne. Exact penalty function algorithm with simple updating of the penalty parameter. *Journal of Optimization Theory and Applications*, 69: 441–467, 1991.
- R. S. Dembo and T. Steihaug. Truncated-Newton methods algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.
- R.S. Dembo, S. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.
- D. den Hertog. *Interior Point Approach to Linear, Quadratic and Convex Programming*. Kluwer Academic Publishers, 1994.
- J.E. Dennis and J. Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Mathematics of Computation*, 28:549–560, 1974.
- J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Non-linear Equations*. Prentice-Hall, 1983.
- G. Di Pillo. Exact penalty methods. In E. Spedicato, editor, *Algorithms for Continuous Optimization: the State of the Art*, pages 1–45. Kluwer Academic Publishers, 1994.
- G Di Pillo, F. Facchinei, and L. Grippo. An RQP algorithm using a differentiable exact penalty function for inequality constrained problems. *Mathematical Programming*, 55:49–68, 1992.
- G. Di Pillo and L. Grippo. A new class of augmented Lagrangians in nonlinear programming. *SIAM Journal on Control and Optimization*, 17:618–628, 1979.

- G. Di Pillo and L. Grippo. A continuously differentiable exact penalty function for nonlinear programming problems with inequality constraints. *SIAM Journal on Control and Optimization*, 23:72–84, 1985.
- G. Di Pillo and L. Grippo. Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization*, 27:1333–1360, 1989.
- G. Di Pillo, L. Grippo, and F. Lampariello. A method for solving equality constrained optimization problems by unconstrained minimization. In K. Iracki, K. Malanowski, and S. Walukiewicz, editors, *Optimization Techniques*, Proc. 9th IFIP Conf., Warsaw, 1979, pages 96–105. Springer-Verlag, 1980.
- G. Di Pillo, L. Grippo, and F. Lampariello. A class of methods for the solution of optimization problems with inequalities. In R. F. Drenick and F. Kozin, editors, *System Modelling and Optimization*, Proc. 10th IFIP Conf., New York, 1981, pages 508–519. Springer-Verlag, 1982.
- G. Di Pillo, L. Grippo, and S. Lucidi. Globally convergent exact penalty algorithms for constrained optimization. In A. Prekopa, J. Szelezsan, and B. Strazicky, editors, *System Modelling and Optimization*, Proc. 12th IFIP Conf., Budapest, 1985, pages 694–703. Springer-Verlag, 1986.
- L. Dixon. Exact penalty function methods in nonlinear programming. Technical Report 103, Numerical Optimization Centre, Hatfield Polytechnic, 1979.
- Y. G. Evtushenko. *Numerical Optimization Techniques*. Optimization Software, Inc., 1985.
- F. Facchinei. Robust quadratic programming algorithm model with global and superlinear convergence properties. *Journal of Optimization Theory and Application*, 92:543–579, 1997.
- M. C. Ferris, S. Lucidi, and M. Roma. Nonmonotone curvilinear stabilization techniques for unconstrained optimization. *Computational Optimization and Applications*, 6:117–136, 1996.
- A.V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, 1968.
- R. E. Fletcher. A class of methods for nonlinear programming with termination and convergence properties. In J. Abadie, editor, *Integer and Nonlinear Programming*, pages 157–173. North-Holland, 1970a.
- R. E. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970b.
- R. E. Fletcher. Numerical experiments with an exact l_1 penalty function method. In O.L. Mangasarian, R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 4*, pages 99–129. Academic Press, 1981.
- R. E. Fletcher. *Practical Methods of Optimization*, (2nd edition). John Wiley & Sons, 1987.
- R. E. Fletcher and M. J. D. Powell. A rapidly convergent descent algorithm for minimization. *Computer Journal*, 6:163–168, 1963.
- R. E. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

- R. M. Freund and S. Mizuno. Interior point methods: current status and future directions. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 441–466. Kluwer Academic Publishers, 2000.
- K. R. Frisch. The logarithmic potential method of convex programming, 1955. Memorandum of May 13, 1955, University Institute of Economics, Oslo.
- U.M. Garcia-Palomares and O.L. Mangasarian. Superlinearly convergent quasi-Newton methods for nonlinearly constrained optimization problems. *Mathematical Programming*, 11:11–13, 1976.
- D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, 2:186–197, 1981.
- F. Giannessi. *Metodi Matematici della Programmazione. Problemi lineari e non lineari*. Pitagora Editrice, 1982.
- J. C. Gilbert and J. Nocedal. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization*, 2:21–42, 1992.
- P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.
- T. Glad and E. Polak. A multiplier method with automatic limitation of penalty growth. *Mathematical Programming*, 17:140–155, 1979.
- Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.
- N. I.M. Gould, S. Lucidi, M. Roma, and P. Toint. Solving the trust-region subproblems using the Lanczos method. *SIAM Journal on Optimization*, 9:504–525, 1999.
- L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton’s method. *SIAM Journal on Numerical Analysis*, 23:707–716, 1986.
- L. Grippo, F. Lampariello, and S. Lucidi. Global convergence and stabilization of unconstrained minimization methods without derivatives. *Journal of Optimization Theory and Applications*, 56:385–406, 1988.
- L. Grippo, F. Lampariello, and S. Lucidi. A truncated Newton method with nonmonotone linesearch for unconstrained optimization. *Journal of Optimization Theory and Applications*, 60:401–419, 1989.
- L. Grippo, F. Lampariello, and S. Lucidi. A class of nonmonotone stabilization methods in unconstrained optimization. *Numerische Mathematik*, 59:779–805, 1991.
- L. Grippo and S. Lucidi. A globally convergent version of the Polak-Ribiere conjugate gradient method. *Mathematical Programming*, 78:375–391, 1997.
- S. P. Han. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.
- S. P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Application*, 22:297–309, 1977.
- M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969.

- M. R. Hestenes. *Optimization Theory. The Finite Dimensional Case*. John Wiley & Sons, 1975.
- R. Horst and P. Pardalos. *Handbook of Global Optimization*. Kluwer Academic Publishers, 1995.
- K. Jahn. *Introduction to the Theory of Nonlinear Optimization*. Springer-Verlag, 1994.
- C. T. Kelley. *Iterative Methods for Optimization*. SIAM, 1999.
- R. M. Lewis, V. Torczon, and M. W. Trosset. Why pattern search works. *Optima: Mathematical Programming Newsletter*, (59), 1998.
- C.-J. Lin and J. J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM Journal on Scientific Computing*, 21:24–45, 1999.
- D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.
- S. Lucidi. Recursive Quadratic Programming algorithm that uses an exact augmented Lagrangian function. *Journal of Optimization Theory and Applications*, 67:227–245, 1990.
- S. Lucidi, L. Palagi, and M. Roma. On some properties of quadratic programs with a convex quadratic constraint. *SIAM Journal on Optimization*, 8:105–122, 1998a.
- S. Lucidi, F. Rochetich, and M. Roma. Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. *SIAM Journal on Optimization*, 8:916–939, 1998b.
- S. Lucidi and M. Roma. Numerical experience with new truncated Newton methods in large scale unconstrained optimization. *Computational Optimization and Applications*, 7:71–87, 1997.
- S. Lucidi and M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization without derivatives (revised version). Technical Report 18-96, DIS, Università di Roma La Sapienza, 1996.
- D. G. Luenberger. *Linear and Nonlinear Programming (2nd edition)*. Addison-Wesley, 1994.
- O. L. Mangasarian. *Nonlinear Programming*. McGraw-Hill, 1969.
- N. Maratos. Exact penalty function algorithms for finite dimensional and control optimization problems, 1978. Ph.D. Thesis, University of London.
- D.Q. Mayne and E. Polak. A superlinearly convergent algorithm for constrained optimization problems. *Mathematical Programming Study*, 16:45–61, 1982.
- G. P. McCormick. *Nonlinear Programming - Theory, Algorithms, and Applications*. John Wiley & Sons, 1983.
- J. J. McKeown, D. Meegan, and D. Sprevak. *An Introduction to Unconstrained Optimization*. IOP Publishing Ltd., 1990.
- A. Miele, E. E. Cragg, R. R. Iyer, and A. V. Levy. Use of the augmented penalty function in mathematical programming, part I and II. *Journal of Optimization Theory and Applications*, 8:115–130 and 131–153, 1971.

- A. Miele, P. Moseley, A. V. Levy, and G. H. Coggins. On the method of multipliers for mathematical programming problems. *Journal of Optimization Theory and Applications*, 10: 1–33, 1972.
- K. Miettinen. *Nonlinear Multi-Objective Optimization*. Kluwer Academic Publishers, 1999.
- M. Minoux. *Programmation Mathématique. Théorie and Algorithms*. Dunod, 1983.
- J. J. Moré and D. C. Sorensen. On the use of direction of negative curvature in a modified Newton direction. *Mathematical Programming*, 16:1–20, 1979.
- J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4:553–572, 1983.
- J. J. Moré and D. C. Sorensen. Newton’s method. In G. H. Golub, editor, *Studies in Numerical Analysis*, vol.24 of MAA Studies in Mathematics, pages 29–82. The Mathematical Association of America, 1984.
- S. G. Nash. Sumt (revisited). *Operations Research*, 46:763–775, 1998.
- S. G. Nash. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics*, 1999. to appear.
- S. G. Nash and J. Nocedal. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM Journal on Optimization*, 1: 358–372, 1991.
- S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.
- Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 1994.
- J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 2:199–242, 1992.
- J. Nocedal. Large Scale Unconstrained Optimization. In A. Watson and I. Duff, editors, *The state of the art in Numerical Analysis*, pages 311–338. Oxford University Press, 1996.
- J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, 1999.
- J. M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems*. Plenum Press, 1988.
- J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.
- E. R. Paniers and A. L. Tits. On combining feasibility, descent and superlinear convergence in inequality constrained optimization. *Mathematical Programming*, 59:261–276, 1993.
- A.L. Peressini, F.E. Sullivan, and J.J. Uhl, Jr. *The Mathematics of Nolinear Programming*. Springer-Verlag, 1988.
- T. Pham Dinh and L. T. Hoai An. D.C. optimization algorithm for solving the trust region subproblem. *SIAM Journal on Optimization*, 8:476–505, 1998.
- J. D. Pinter. *Global Optimization in Action: Continuous and Lipschitz Optimization – Algorithms, Implementations and Applications*. Kluwer Academic Publishers, 1996.

- E. Polak. *Optimization: Algorithms and Consistent Approximations*. Springer Verlag, 1997.
- E. Polak and G. Ribière. Note sur la convergence de méthodes de directions conjuguées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 16:35–43, 1969.
- B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., 1987.
- M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. E. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, 1969.
- M. J. D. Powell. On the convergence of the variable metric algorithm. *Journal of the Institute of Mathematics and its Applications*, 7:21–36, 1971.
- M. J. D. Powell. Convergence properties of a class of minimization algorithms. In O.L. Mangasarian, R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 2*. Academic Press, 1975.
- M. J. D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In R. W. Cottle and C. E. Lemke, editors, *Nonlinear Programming, SIAM-AMS Proceedings*. SIAM Publications, 1976.
- M. J. D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming*, 14:224–248, 1978a.
- M. J. D. Powell. The convergence of variable metric methods for nonlinear constrained optimization calculation. In O.L. Mangasarian, R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 3*. Academic Press, 1978b.
- M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, Dundee 1977*, pages 144–157. Springer-Verlag, 1978c.
- M. J. D. Powell. Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, 28:487–500, 1986.
- M. J. D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7: 287–336, 1998.
- M. J. D. Powell and Y. Yuan. A recursive quadratic programming algorithm that uses differentiable exact penalty functions. *Mathematical Programming*, 35:265–278, 1986.
- T. Rapsáck. *Smooth Nonlinear Optimization in \mathbb{R}^n* . Kluwer Academic Publishers, 1997.
- M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problems. *SIAM Journal on Optimization*, 7:26–33, 1997.
- F. Rendl and H. Wolkowicz. A semidefinite framework to trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77:273–299, 1997.
- R. T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12:555–562, 1973.
- C. Roos, T. Terlaky, and J.-Ph. Vial. *Interior Point Approach to Linear Optimization: Theory and Algorithms*. John Wiley & Sons, 1997.
- J.B. Rosen. The gradient projection method for nonlinear programming, part I: linear constraints. *SIAM Journal on Applied Mathematics*, 8:181–217, 1960.

- J.B. Rosen. The gradient projection method for nonlinear programming, part II: nonlinear constraints. *SIAM Journal on Applied Mathematics*, 9:514–553, 1961.
- B. Rustem. *Algorithms for Nonlinear Programming and Multiple-Objective Decisions*. John Wiley & Sons, 1998.
- D.F. Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 27:647–656, 1970.
- D. C. Sorensen. Newton’s method with a model trust region modification. *SIAM Journal on Scientific and Statistical Computing*, 19:409–427, 1982.
- D. C. Sorensen. Minimization of a large-scale quadratic function subject to an ellipsoidal constraint. *SIAM Journal on Optimization*, 7:141 – 161, 1997.
- E. Spedicato. *Algorithms for Continuous Optimization: The State of the Art*. Kluwer Academic Publishers, 1994.
- T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20:626–637, 1983.
- P. Toint. Towards an efficient sparsity exploiting Newton methods for minimization. In I. S. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–87. Academic Press, 1981.
- V. Torczon. Pattern search methods for nonlinear optimization. *SIAG/OPT Views-and-News*, (6), 1995.
- V. Torczon. On the convergence of pattern search methods. *SIAM Journal on Optimization*, 7:1–25, 1997.
- A. Törn and A. Žilinskas. *Global Optimization*. Springer-Verlag, 1989.
- R.B. Wilson. A simplicial algorithm for concave programming, 1963. Ph.D. Thesis, Harvard University.
- M. A. Wolfe. *Numerical Methods for Unconstrained Optimization. An Introduction*. Van Nostrand, 1978.
- P. Wolfe. Methods of nonlinear programming. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 67–86. McGraw-Hill, 1963.
- S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.
- Y. Ye. *Interior Point Algorithms. Theory and Analysis*. Wiley-Interscience, 1997.
- W. I. Zangwill. *Nonlinear Programming - A Unified Approach*. Prentice Hall, 1969.
- W.I. Zangwill. Nonlinear programming via penalty functions. *Management Science*, 13:344–358, 1967.
- G. Zoutendijk. *Methods of Feasible Directions*. Elsevier, 1960.