Non-linear optimization and large-scale problems

Jean Charles Gilbert

Nonlinear optimization and large-scale problems[1]

Jean Charles GILBERT
INRIA, B. P. 105, F-78153 Le Chesnay Cedex
May 1989 (revised in June 1990)

Abstract. *This paper recalls some basic concepts of unconstrained optimization techniques and reviews some useful methods for solving large-scale problems: conjugate gradient methods without critical line-searches, limited memory quasi-Newton methods and the partitioned quasi-Newton method. We shall discuss their efficiency, their convergence properties and their respective advantages and disadvantages.*

Key words: conjugate gradient, large-scale optimization, limited memory quasi-Newton method, partitioned quasi-Newton method, unconstrained optimization.

1 Introduction

Optimization problems are frequent in engineering, physics (minimization of energy), economics (minimization of costs) and mathematics (variational problems, least-squares methods), etc. When they are related to real-life problems, very often, a large number of variables is involved. Then, we talk about large-scale optimization.

The notion of large-scale problems does not depend only on the number of independent parameters that describe the problem. Actually, it would be more suitable to say that a large-scale problem is a problem hard to solve by "traditional" techniques. In this respect, the notion of large-scale problem is time dependent: the increase in storage capacities and calculation speed of today's computers has raised the limit separating small and large-scale problems. This notion is also dependent on the nature of the problem: a quadratic function whose Hessian has clustered eigenvalues is easier to minimize than a general nonlinear function. To fix the ideas, however, we may say that a problem without any particular structure is large today if its number n of variables is greater than 500. To fix this number, we considered mainly the updating cost and memory "clutter" of $O(n^2)$ elements of storage.

Methods that are good for problems of small or medium scale need to use information on the curvature of the function to minimize, i.e., on its Hessian matrix. For large-scale problems, such methods become unusable because they require too much storage (this is the case with the standard Newton method). They can also become inefficient, because they

[1] Presented at the second conference on Mathematics for Engineers: "Mathematical Programming: a Tool for Engineers", May 17-19, 1989, Mons (Belgium). To appear in Engineering Optimization.

require too many iterations to collect the proper information (this is the case with quasi-Newton (QN) methods). Therefore, one possibility is to use methods that are less efficient but that require fewer locations in memory. The conjugate gradient methods are of this type. They are considered in Section 3. Another possibility is to adapt fast methods to large-scale problems. This is the case of limited memory QN methods (discussed in Section 4) and the case of the partitioned QN method (examined in Section 5).

With these three classes of methods, we shall not cover all the possibilities. In particular, almost nothing will be said on the discrete Newton method, which can be good alternative (references are given below). Also, we shall limit our discussion to unconstrained optimization and to descent-direction methods.

To conclude this introduction, let us mention some optimization books and review papers. The books by Gill, Murray and Wright (1981), by McCormick (1983), by Minoux (1983), also available in English, by Dennis and Schnabel (1983) and by Fletcher (1987) are now classical, see also Lemaréchal (1989). They are recommended as introduction to optimization techniques. The book by Bertsekas (1982) is also recommended but is more specialized. The book by Gruver and Sachs (1980) deals with optimization algorithms for optimal control. The book by Bertsekas and Tsitsiklis (1989) describes some optimization algorithms that are suitable for parallelization. The references [10], [27], [42], [3], [23], [34], [38] and [12] are collections of papers giving the state of the art in optimization. Finally, let us mention the review paper by Toint (1986) on large-scale optimization, which contains an important bibliography.

2 Basic concepts

Let $\mathbb{R}^n$ be equipped with a scalar product $\langle \cdot, \cdot \rangle$ and its associated norm $| \cdot |$. We shall use the following matrix norm on $\mathbb{R}^n$: $||A|| := \sup\{|Au| \, : \, |u| = 1\}$.

Given a smooth function $f : \Omega \to \mathbb{R}$, defined on an open set Ω of $\mathbb{R}^n$, the unconstrained minimization problem consists in finding a point $x_* \in \Omega$, such that

$$f(x_*) \, \leq \, f(x), \; \forall \, x \in N(x_*) \, , \tag{1}$$

where $N(x_*)$ denotes some neighborhood of x_*. A point x_* satisfying the above inequalities is called a *local minimizer* of f. If (1) holds for $N(x_*) = \Omega$, then x_* is called a *global minimizer* of f on Ω. Finding a global minimizer is a hard task and for this subject, we refer the reader to the collections of papers by Dixon and Szegö (1975, 1978); see also [34]. The algorithms considered in this paper are only able to find local minimizers.

If x_* is a local minimizer, it satisfies the first order optimality conditions:

$$f'(x_*) \cdot h \, = \, 0, \; \forall \, h \in \mathbb{R}^n \, ,$$

where $f'(x_*)$ is the Fréchet derivative of f at x_*. These conditions can be rewritten as follows:

$$g(x_*) \, = \, 0 \, , \tag{2}$$

where $g(x) \, := \, \nabla f(x)$ denotes the *gradient* of f at x. It is important to keep in mind that the gradient depends on the scalar product chosen on $\mathbb{R}^n$. It is indeed defined by means of the Riesz theorem:

$$\exists \, g(x) \in \mathbb{R}^n, \text{unique} \; : \;\; \langle g(x), h \rangle \, = \, f'(x) \cdot h, \; \forall \, h \in \mathbb{R}^n \, .$$

If the Euclidean scalar product is used, $\langle u, v\rangle = \sum_{i=1}^{n} u_i v_i$, then $g_i(x)$ is the i-th partial derivative of f at x. A point x_* satisfying (2) is called a *stationary point* of f. A minimizer is a stationary point and the converse is true for convex function.

Here, we have to be still more modest as to what the algorithms described in this paper can do. In general, they can only find stationary points. This is due to the fact that they will use only the value of f and the value of its gradient. If for some reasons, a local minimizer has to be found (and not only a stationary point) and if there are many stationary points that are not minimizers, something more has to be done. A good idea is to use second order information to locate directions of negative curvature. Along this line, see the papers by Gill and Murray (1974), McCormick (1977), Moré and Sorensen (1979) and Goldfarb (1980).

Finding a stationary point of a function is necessarily an iterative process. Indeed, in the simplified case where $n = 1$ and f is a polynomial of degree $p + 1$, the problem is equivalent (by the optimality condition (2)) to finding the roots of a polynomial of degree p. Since Galois, it is known that this problem cannot be solved exactly by using simple arithmetic operations, as soon as $p \geq 5$. Therefore, the only thing that is reasonable to expect from an optimization method is that it generates a sequence $\{x_k\}_{k\geq 1}$ of points that converges to a solution.

An important class of methods generating sequences of approximations of a solution is formed by the *descent-direction methods*. They proceed as follows. It is assumed that some approximation x_k is known at the beginning of iteration k, $k \geq 1$ (x_1 has to be given at the beginning of the run). Then, an iteration consists of calculating a new approximation x_{k+1} that is better than the previous one, in the sense that

$$f(x_{k+1}) < f(x_k)\,. \tag{3}$$

To obtain this *descent property*, the first stage consists in choosing a *descent direction* at x_k, i.e., a direction d_k verifying

$$f'(x_k) \cdot d_k < 0\,. \tag{4}$$

The second stage consists in selecting a positive step-size ρ_k along d_k and taking

$$x_{k+1} = x_k + \rho_k\, d_k$$

such that x_{k+1} satisfies, at least, the descent property (3). This stage is called the *line-search*. Let us summarize this method as follows.

The descent-direction algorithm:
1. choose $x_1 \in \Omega$;
2. $k := 1$;
3. **until** convergence **do {**
3.1. choose a descent direction d_k;
3.2. select a positive step-size ρ_k along d_k such that (3) holds;
3.3. $x_{k+1} := x_k + \rho_k\, d_k$;
3.4. $k := k + 1$
3. **}**

The way of calculating the descent direction characterizes a descent-direction method. For example, when $g_k := g(x_k) \neq 0$, an obvious descent direction is:

$$d_k = -g_k\,. \tag{5}$$

The method that uses this descent direction at each iteration is the *gradient method*. More generally, if H_k is a positive definite matrix of order n,

$$d_k = -H_k\, g_k \tag{6}$$

will be a descent direction. Many practical algorithms for minimizing a function have this form of descent direction: the Newton method (near a solution), the BFGS method (or more generally, algorithms in the family of quasi-Newton methods) and even conjugate gradient methods, as we shall see in Section 4.

The descent condition (4) implies that for sufficiently small positive step-size ρ, we have $f(x_k + \rho d_k) < f(x_k)$. Therefore, to satisfy (3), a step-size selection procedure could simply choose a small value for ρ. This is not satisfactory, however, because examples can be given for which taking arbitrarily small positive step-sizes forces a descent algorithm to generate a sequence $\{x_k\}_{k\geq 1}$ that converges to a nonstationary point (*false convergence*). This phenomenon is dangerous and hopefully, it can be avoided in several ways. We mention two of them.

A first idea is to try to get the maximal decrease of f in the direction d_k, in other words to minimize the function $\xi_k(\rho) := f(x_k + \rho d_k)$ for positive ρ. We shall say that a step-size selection procedure realizes an *optimal line-search* if it selects an *optimal step-size*, that is to say a positive step-size ρ_k such that $x_{k+1} = x_k + \rho_k d_k$ satisfies

$$f(x_{k+1}) \leq f(x_k + \bar{\rho}_k\, d_k)\,, \tag{7}$$

where $\bar{\rho}_k$ is the smallest positive stationary point of the function ξ_k. A step-size that is a stationary point of ξ_k will be said to be *critical* and is obtained by a *critical line-search*. The first local minimizer and the global minimizer of $\{\, \xi_k(\rho) \,:\, \rho > 0 \,\}$, if they exist, are optimal and critical. Actually, the optimal and critical line-search strategies have two drawbacks. First, the step-size $\bar{\rho}_k$ may not exist at some iteration although problem (1) has a solution. Secondly, finding a stationary point of ξ_k is a nonlinear minimization problem in itself and this cannot be solved exactly, except in particular cases (for instance, when f is quadratic). Therefore, although this procedure is theoretically suitable (see Theorem 1 below), it is useless in practice.

If an optimal step-size is not computable, it is possible to calculate almost optimal step-sizes, in a sense to precise. In fact, the convergence analysis of descent-direction methods has shown that to be acceptable a step-size has to define a point x_{k+1} satisfying some conditions. First, the descent property (3) is not strong enough: it does not force a sufficient decrease of the objective function. Instead, the following inequality with $\omega_1 \in \,]\,0,\, 1[$ is suitable:

$$f(x_{k+1}) \leq f(x_k) + \rho_k\, \omega_1 \langle g_k, d_k \rangle\,. \tag{8}$$

It is easy to verify that this inequality is always satisfied for small positive ρ_k. If this is comforting, it means also that it does not prevent the step-size from being too small. As we said, this can lead to false convergence. Therefore, another condition is needed. For example, the following inequality

$$\langle g(x_{k+1}), d_k \rangle \;\geq\; \omega_2 \langle g_k, d_k \rangle \tag{9}$$

with $\omega_2 \in \,]\,\omega_1,\, 1[$ can be used. Remark that it does not accept small step-sizes and that in particular, $\rho_k = 0$ is rejected. Actually, this inequality expresses that the slope of the function ξ_k at ρ_k has to be sufficiently larger than the slope at the origin. Conditions (8) and (9) are

called the *Wolfe conditions* and a point x_{k+1} satisfying them is called a *Wolfe point*. It can be proved (see Wolfe (1969)) that if f is bounded below and if $0 < \omega_1 < \omega_2 < 1$, it is always possible to find a Wolfe point. Typical values for the parameters are $\omega_1 = 10^{-4} \dots 10^{-2}$ and $\omega_2 = 0.5 \dots 0.99$. In any case, it is healthy to take $\omega_1 < 0.5$, so that the step-size minimizing the quadratic approximation of ξ_k will be accepted by the inequality (8). Calculating a Wolfe point requires a specific algorithm. The one proposed by Lemaréchal (1981) has been proved to find a Wolfe point in a finite number of trials. For some algorithms, we shall need a step-size satisfying the *strong Wolfe conditions*, that is to say, satisfying (8) and an inequality stronger than (9), namely

$$|\langle g(x_{k+1}), d_k \rangle| \leq \omega_2 |\langle g_k, d_k \rangle| . \tag{10}$$

An algorithm for finding a strong Wolfe point can be found in [2] and [33].

The preceding two line-search techniques allow us to obtain the following result that can be derived from the papers by Zoutendijk (1970) and Wolfe (1969 and 1971). Before stating the theorem, let us specify some notation. We shall denote by θ_k the angle between d_k and -g_k, that is

$$\cos\theta_k := -\frac{\langle g_k, d_k \rangle}{|g_k|\,|d_k|} .$$

If $\{\epsilon_k\}_{k\geq 1}$ is a sequence of real numbers, the notation "$\liminf \epsilon_k$" will mean "$\lim_{k\to\infty} \inf_{1\leq i\leq k} \epsilon_i$", which differs from the notation in topology.

Theorem 1 *Let f be a differentiable function defined on the open set Ω, with a Lipschitz continuous derivative. Suppose that a descent-direction method generates a sequence of points $\{x_k\}_{k\geq 1}$ in Ω and a sequence of directions $\{d_k\}_{k\geq 1}$ verifying (4). Suppose also that at each iteration the line-search procedure finds either an optimal point satisfying (7) or a Wolfe point satisfying (8) and (9). Suppose finally that $\{f(x_k)\}_{k\geq 1}$ is bounded below. Then, the following conclusions hold:*

(i) $\sum_{k\geq 1} |g_k|^2 \cos^2\theta_k$ *is convergent;*
(ii) $|g_k| \cos\theta_k \to 0$*;*
(iii) *if* $\sum_{k\geq 1} \cos^2\theta_k$ *is divergent, then* $\liminf |g_k| = 0$.

The conclusions (ii) and (iii) are clear consequences of the conclusion (i).

The theorem says nothing on the convergence of the sequence of points $\{x_k\}_{k\geq 1}$. Only the sequence of gradients is concerned, although it is not claimed that g_k converges to zero. Actually, Theorem 1 expresses the contribution of the line-search procedure to force convergence. Now, to make its conclusions useful, some more information concerning the way the descent directions d_k are chosen is needed. For instance, if the angle θ_k remains bounded away from $\pm\pi/2$, then $\cos\theta_k$ remains bounded away from 0 and the conclusion (ii) of Theorem 1 implies that the sequence of gradients converges to zero and therefore that any limit point of $\{x_k\}_{k\geq 1}$ is stationary. This is the case for the gradient method, because $\theta_k = 0, \forall k \geq 1$. This is also the case for the method (6), if the condition numbers $\kappa(H_k)$ of the positive definite matrices H_k remain bounded; that is, if there exists a positive constant $\bar{\kappa}$ such that

$$\kappa(H_k) := ||H_k||\,||H_k^{-1}|| \leq \bar{\kappa}, \quad \forall k \geq 1.$$

Indeed, we have $\cos\theta_k \geq \kappa(H_k)^{-1}$.

These remarks indicate that to be sure to have a convergent algorithm, it is enough to use method (6) with constant matrices H_k (the identity matrix, for instance) or with matrices

having bounded condition number. Our task is not finished, however. Indeed, such directions can be very poor descent directions in the sense that the decrease of the objective function may be small along them. Therefore, in the subsequent sections, we shall play the dangerous game of trying to define "good" descent directions by taking information from the function with the risk of violating the conditions of convergence stated in Theorem 1. For some of the methods below, it is still not clear whether they can fail to converge when they aim at minimizing general functions.

3 Conjugate gradient methods without optimal line-searches

The conjugate gradient (CG) method was introduced to solve linear systems of equations [22] and later to minimize strongly convex quadratic functions [14]. For these problems, the method has the outstanding feature to terminate in at most n iterations. This is a dramatic improvement in comparison with the gradient method, which usually requires an infinite number of iterations to converge.

The basic idea of the CG method is to generate a set of conjugate directions, i.e., directions that are orthogonal with respect to the scalar product induced by the Hessian matrix A of f. As a result, f is minimized on the larger and larger affine subspace spanned by the successive directions, the *Krylov subspace*. When the first direction is $-g_1$, the conjugacy of the directions can be obtained by the following simple formula:

$$d_k = -g_k + \beta_k^{\text{HS}}\, d_{k-1}\,,$$

where β_k^{HS} is the Hestenes-Stiefel beta (see [22]):

$$\beta_k^{\text{HS}} := \frac{\langle y_{k-1}, g_k\rangle}{\langle y_{k-1}, d_{k-1}\rangle}\,.$$

We note $y_k := g_{k+1} - g_k$ and $s_k := x_{k+1} - x_k$. To have conjugacy, it is necessary to do optimal line-searches, which does not pose any difficulty in case of quadratic functions. It is worth noting that if the CG method is better than the gradient method, it also uses information from previous iterates: d_{k-1} is used in the definition of d_k. This feature is typical of efficient algorithms.

The CG algorithm for quadratic functions:
1. choose a point $x_1 \in \Omega$;
2. $d_1 := -g_1$; $k := 1$;
3. **repeat {**
3.1. $\rho_k := -\langle g_k, d_k\rangle / \langle Ad_k, d_k\rangle$; /* the optimal step-size */
3.2. $x_{k+1} := x_k + \rho_k\, d_k$;
3.3. **if** convergence **then stop**;
3.4. $d_{k+1} := -g_{k+1} + \beta_{k+1}^{\text{HS}}\, d_k$;
3.5. $k := k + 1$
3. **}**

Usually, real-life large-scale models are not quadratic. The question is therefore to know whether, for minimizing a general nonlinear function, it still makes sense to use a CG-like

method defined for some scalar β_k by

$$\left\{ \begin{array}{l} d_1 = -g_1 \\ d_k = -g_k + \beta_k \, d_{k-1} \, , \text{ for } k \geq 2 \, . \end{array} \right. \tag{11}$$

Note that for non quadratic functions, it is still possible to define a local notion of conjugacy by using the average Hessian of f:

$$\bar{B}_k = \int_0^1 \nabla^2 f(x_k + t s_k) \, dt \, .$$

Indeed, because $y_k = \bar{B}_k s_k$, equation (11) with $\beta_k = \beta_k^{\rm HS}$ implies that $\langle \bar{B}_k \, d_k, d_{k+1} \rangle = 0$ for $k \geq 1$. Therefore, d_k and d_{k+1} are conjugate with respect to $\bar{B}_k$. However, it does not seem that this concept is very helpful in the non quadratic case and therefore, on this basis only, it is hard to claim that the HS beta is better than any other beta.

Another point is to know whether the directions defined by (11) are downhill: do we have $\langle g_k, d_k \rangle$ negative? As $\langle g_k, d_k \rangle = -|g_k|^2 + \beta_k \langle d_{k-1}, g_k \rangle$, an obvious way of getting descent directions is to do critical line-searches: $\langle d_{k-1}, g_k \rangle = 0$. We have said that such strategy is not realistic. Hopefully, it can be avoided. The following theorem (see [16, Theorem 3.2]) is an extension of a result of Al-Baali (1985). It shows that provided $|\beta_k|$ is not too large, a line-search procedure realizing the strong Wolfe conditions automatically ensures descent of the direction d_k. Furthermore, global convergence can be proved. Note that this result strongly relies on the fact that the first search direction is $-g_1$. In this theorem, the Fletcher-Reeves beta is used to control the magnitude of β_k. It is defined by (see [14]):

$$\beta_k^{\rm FR} := \frac{|g_k|^2}{|g_{k-1}|^2} \, .$$

For quadratic functions and optimal line-searches, $\beta_k^{\rm HS} = \beta_k^{\rm FR}$.

Theorem 2 *Let f be a twice continuously differentiable function defined on the open set $\Omega = I\!R^n$ and let x_1 be a point in Ω. Suppose that the level set $\{ \, x \in \Omega \; : \; f(x) \leq f(x_1) \, \}$ is bounded. If the nonlinear* CG *method (11) with $|\beta_k| \leq \beta_k^{\rm FR}$ starts from x_1 and uses the strong Wolfe line-search with $0 < \omega_1 < \omega_2 < 1/2$, then the directions d_k are downhill and* $\liminf |g_k| = 0$.

This result could let think that the CG method (11) with $\beta_k = \beta_k^{\rm FR}$ and the strong Wolfe line-search is a good algorithm. Unfortunately, it is not the case. The point is that this theorem only ensures the global convergence of the method, i.e., the convergence from any starting point, but not its fast convergence. In fact, numerical experiments with the Fletcher-Reeves formula have shown that the method can converge very slowly. A partial clarification of this phenomenon has been given by Powell (1977, 1985) who gave a reason why the Polak-Ribière beta, defined by (see [39]):

$$\beta_k^{\rm PR} := \frac{\langle y_{k-1}, g_k \rangle}{|g_{k-1}|^2} \, ,$$

should by used instead of $\beta_k^{\rm FR}$ (for critical line-searches, $\beta_k^{\rm HS} = \beta_k^{\rm PR}$). Numerical experiments have confirmed the superiority of the PR beta over the FR beta.

The misfortune is that the PR method is not as safe as the FR method. If it can be proved that the method is convergent for convex functions (see [39]), Powell (1984) has given an example of a nonconvex function where the PR method diverges: the generated sequence cycles and each of its limit points is nonstationary! Two globally convergent remedies to this situation have been proposed in [16]. The first one is inspired by Theorem 2. It consists of using the PR beta only when it lies in the interval $[-\beta_k^{\rm FR}, +\beta_k^{\rm FR}]$. Specifically,

$$\textbf{Method PR|FR}: \quad \beta_k = \begin{cases} -\beta_k^{\rm FR} & \text{if} \quad \beta_k^{\rm PR} < -\beta_k^{\rm FR} \\ \beta_k^{\rm PR} & \text{if} \quad |\beta_k^{\rm PR}| \leq \beta_k^{\rm FR} \\ \beta_k^{\rm FR} & \text{if} \quad \beta_k^{\rm PR} > \beta_k^{\rm FR}. \end{cases}$$

This strategy avoids one of the main disadvantages of the FR method, as discussed in [16].

Another globally convergent remedy consists of restarting the PR method in the opposite direction of the gradient, each time $\beta_k^{\rm PR}$ is negative. Synthetically, this corresponds to the following choice of β_k:

$$\textbf{Method PR}^+: \quad \beta_k = \max\{0, \beta_k^{\rm PR}\}.$$

For this method, a specific line-search assuring descent of the directions has to be implemented.

FR	PR|FR	PR	PR$^+$
> 4.07	1.55	1.02	1.00

Table 1: Comparison of the methods FR, PR|FR, PR and PR$^+$.

Table 1 allows us to compare the performance of several CG methods: FR, PR, PR|FR and PR$^+$. The results come from tests made in [16] on a collection of 26 large-scale problems with a number of variables between 100 and 10000. The table gives the total number of function/gradient calls used for all the test-problems divided by the 18792 calls required by method PR$^+$. The sign $>$ means that the method was stopped on some test-problems because the number of function/gradient calls exceeds 9999. The actual number should therefore be greater than the one given in the table. Methods PR and PR$^+$ appear to be the best one, while the FR method is on average more than four times slower than method PR$^+$!

4 Limited memory quasi-Newton methods

4.1 The Newton method

The Newton method plays a major part in optimization. It is obtained by considering the local quadratic model of the function given by a second order Taylor expansion of f about the current point x_k:

$$f(x) = f(x_k + d) = m_k(d) + o(|d|^2),$$

$$m_k(d) = f(x_k) + \langle g_k, d\rangle + \frac{1}{2}\langle \nabla^2 f(x_k)\, d, d\rangle,$$

where $\nabla^2 f(x_k)$ denotes the Hessian matrix of f at x_k. This matrix also depends on the scalar product. The displacement of the method is obtained by minimizing the model m_k instead

of f. When $\nabla^2 f(x_k)$ is positive definite, this gives:

$$d_k = -\nabla^2 f(x_k)^{-1} g_k \,. \tag{12}$$

The direction has the form (6) and it is a descent direction when $\nabla^2 f(x_k)$ is positive definite. This is the case when x_k is close to a solution x_* satisfying the second order sufficient conditions of optimality, i.e., with $\nabla^2 f(x_*)$ positive definite.

An attractive feature of Newton's method is its rate of convergence. If the function f is sufficiently smooth and if x_1 is close enough to x_*, then the method generates a *quadratically convergent* sequence $\{x_k\}_{k\geq 1}$, that is

$$||x_{k+1} - x_*|| \leq C\, ||x_k - x_*||^2, \; \forall k \geq 1 \,,$$

for some positive constant C. This property implies that if x_k is close enough to x_*, its number of correct significant digits grows very quickly. Quadratic convergence is typical of algorithms using the second order information contained in the Hessian of f.

If convergence occurs with the Newton method when the initial point x_1 is close to a solution, divergence is often observed if the guess is not good enough. This drawback of the method can be tempered by line-search techniques as described in the introduction or by a trust region approach (see the survey of the topic by Moré (1983)). Other difficulties may occur. For example, when the Hessian is not positive definite, the Newton direction may no longer be downhill. One remedy to this situation is to modify the Hessian in equation (12) to obtain a positive definite matrix: one of the most effective way of doing this is due to Gill and Murray (1974). Another remedy is to solve the linear system (12) within a trust region.

Let us come back to our main concern: large-scale optimization. Can we use the Newton method in this context? The stumbling block is, of course, the memory requirement to store the Hessian matrix. The basic idea is to solve the linear system (12) by a CG method when the Hessian is positive definite or by a Lanczos tridiagonalization otherwise and whenever the quantity $\nabla^2 f(x_k)\, v$ is required, it is evaluated by finite differences:

$$\nabla^2 f(x_k)\, v \simeq \frac{\nabla f(x_k + tv) - \nabla f(x_k)}{t} \,,$$

for some small positive t. This approach is known as the *discrete Newton method*, see the papers by Dembo and Steihaug (1983) and by O'Leary (1983).

4.2 The quasi-Newton methods

The Newton method requires the calculation of the Hessian matrix of f, which can be in certain cases, either too expensive or impossible. This is precisely the domain where the quasi-Newton (QN) methods can be useful. See the review paper by Dennis and Moré (1977).

The descent direction of a QN method has the form (6), where H_k is a matrix that is updated at each iteration by some formula. Hence, the method generates two sequences: a sequence of points $\{x_k\}_{k\geq 1}$ and a sequence of matrices $\{H_k\}_{k\geq 1}$. At the beginning of the run, the algorithm is given a point x_1 *and* a matrix H_1. Often, H_1 is the identity matrix (we shall give better choices later) or a diagonal matrix, like the diagonal of the Hessian $\nabla^2 f(x_1)$, when it is available.

The general scheme of a QN method is the following.

The QN algorithm:

1. choose $x_1 \in \Omega$ and a positive definite matrix H_1;
2. $k := 1$;
3. **until** convergence **do {**
3.1. $d_k := -H_k\, g_k$;
3.2. select a step-size ρ_k by the Wolfe procedure (8) and (9);
3.3. $x_{k+1} := x_k + \rho_k\, d_k$;
3.4. calculate H_{k+1};
3.5. $k := k + 1$
3. **}**

The only thing to examine in this algorithm is the way the matrix H_k is updated. If we set $B_k = H_k^{-1}$ (we shall see how to maintain H_k positive definite, hence nonsingular), we see that B_k plays the part of $\nabla^2 f(x_k)$ in Newton's method. On the other hand, with $s_k := x_{k+1} - x_k$ and $y_k := g_{k+1} - g_k$, we have by Taylor's theorem:

$$y_k = \left(\int_0^1 \nabla^2 f(x_k + t s_k)\, dt \right)\, s_k\,.$$

These two remarks show that it makes sense to impose the matrix B_{k+1}, which has to be calculated at the end of iteration k, to satisfy:

$$y_k = B_{k+1}\, s_k\,.$$

This equation is called the *secant* or *quasi-Newton equation*. If $n \geq 2$, the equation does not determine B_{k+1} completely and some other conditions can be imposed. Generally, they make B_{k+1} keeping information from B_k. There is today a wide consensus that one of the best update formulae for minimization is the BFGS update. To write this formula, it is convenient to use the following *tensor product* of two vectors u and v of $\mathbb{R}^n$: $u \otimes v$ is the rank one matrix defined by $(u \otimes v)h = \langle v, h\rangle u\,, \forall h \in \mathbb{R}^n$. If the Euclidean scalar product is used, $u \otimes v$ is the matrix uv^T. See the appendix in [15] for a derivation of the BFGS formula in this setting, which differs from the usual one by the use of a general scalar product. The BFGS *formula* can be written like this:

$$B_{k+1} = B_k + \frac{y_k \otimes y_k}{\langle y_k, s_k\rangle} - \frac{(B_k s_k) \otimes (B_k s_k)}{\langle B_k s_k, s_k\rangle} =: U(B_k,\, y_k,\, s_k)\,. \tag{13}$$

Note that the inverse matrices H_k can also be updated by using the *inverse* BFGS *formula*:

$$H_{k+1} = \left(I - \frac{s_k \otimes y_k}{\langle y_k, s_k\rangle}\right) H_k \left(I - \frac{y_k \otimes s_k}{\langle y_k, s_k\rangle}\right) + \frac{s_k \otimes s_k}{\langle y_k, s_k\rangle} =: \bar{U}(H_k,\, y_k,\, s_k)\,. \tag{14}$$

Precisely, if $H_k = B_k^{-1}$, $B_{k+1} = U(B_k,\, y_k,\, s_k)$ and $H_{k+1} = \bar{U}(H_k,\, y_k,\, s_k)$, then $H_{k+1} = B_{k+1}^{-1}$.

Formulae (13) and (14) have the nice property of transmitting the positive definiteness of B_k to B_{k+1} (respectively of H_k to H_{k+1}), if and only if $\langle y_k, s_k\rangle$ is positive. Therefore to have H_k positive definite, it will suffice to take H_1 positive definite, to update the matrices by the inverse BFGS formula and to ensure the positivity of $\langle y_k, s_k\rangle$ at each iteration. Note that for strictly convex functions, $\langle y_k, s_k\rangle$ is automatically positive and that for nonconvex functions, the second inequality (9) of the Wolfe line-search implies the positivity of $\langle y_k, s_k\rangle$. This important remark makes the Wolfe step-size selection procedure the most suitable for the QN framework in optimization. It justifies also Statement 3.2 of the QN algorithm.

For problems having small or medium dimension, the BFGS method is very efficient. It is globally convergent for convex problems (see [40]) and for general functions, when convergence occurs, it is fast, namely *superlinearly convergent*. This means that

$$||x_{k+1} - x_*||/||x_k - x_*|| \to 0\,.$$

Although, this is less good than the quadratic rate of convergence of the Newton method, it is generally satisfactory in practice.

4.3 The limited memory quasi-Newton methods

There are many proposals to adapt QN methods to large-scale problems and many of them fit into the following framework.

Although n may be large, the initial matrix H_1 takes generally little space in memory: it is most commonly a positive multiple of the identity matrix. On the other hand, H_k is formed from H_1 and $k-1$ couples $\{(y_i,\, s_i)\}_{1\le i<k}$ by using formula (14). Therefore, to take less storage, one can think of storing these elements instead of H_k and computing $H_k\, g_k$ by an appropriate algorithm. Of course, when the number of iterations increases, these pieces of information become more and more cumbersome in memory and we must get rid of some of the couples $(y_i,\, s_i)$. A method will be called an *m-storage* QN *method* if only m of these couples are used to form H_k from a *starting matrix* H_k^0. In this type of method, the inverse update formula (14) is preferable to the direct formula (13), because the inversion of B_k may be problematic.

The most famous limited memory methods can be seen from the preceding point of view: see [15] for the details. For example, the algorithm CONMIN of Shanno and Phua (1976) is a 2-storage QN method (see also [45]). The algorithm of Buckley and LeNir (1983) is an m-storage QN method where m varies with the iteration index. Finally, the method proposed by Nocedal (1980) is also of this type. We describe it in more detail because we feel it is important.

Nocedal proposed to use always the last m couples (y, s), i.e., $\{(y_i, s_i)\}_{k-m\le i\le k-1}$, to form the matrix H_k: going from iteration k to iteration $k+1$, the couple (y_{k-m}, s_{k-m}) is discarded and cyclically replaced by the new couple (y_k, s_k). Therefore, H_k is always formed from the most recent information and, as a result, it should be more appropriate.

Description of Nocedal's proposal to calculate H_k:
1. choose a positive definite matrix H_k^0;
2. **for** $i := 0$ **to** $m-1$ **do** $H_k^{i+1} := \bar{U}(H_k^i,\, y_{k-m+i},\, s_{k-m+i})$;
3. $H_k := H_k^m$;

Of course, this scheme is formal: in practice, the matrices H_k^i and H_k are not stored (they are too large). Instead, just H_k^0 and the couples $\{(y_i, s_i)\}_{k-m\le i\le k-1}$ are kept in memory and an elegant algorithm, described in [35], is provided to compute $H_k\, g_k$.

The above description of Nocedal's proposal outlines the need of a choice for the starting matrices H_k^0. Contrary to the standard QN methods where this choice has to be made only once at the beginning of a run, here it has to be made at each iteration. Its role is therefore crucial. One of the aims of the work [15] is to examine several possible starting matrices. From this study, two choices emerge.

A first choice is simply to take a judicious multiple of the identity matrix:

$$H_k^0 = \delta_{k-1}\, I\ , \quad \delta_{k-1} := \frac{\langle y_{k-1}, s_{k-1}\rangle}{|y_{k-1}|^2}\ . \tag{15}$$

The factor δ_{k-1} is known as the *Oren-Spedicato factor* (see [37]) and aims at giving H_k^0 a good scaling. Some justifications for using this factor are given in [15].

The other possibility is to take a diagonal matrix and to update it at each iteration. In this case, the algorithm generates and stores two sequences, $\{x_k\}_{k\geq 1}$ and $\{H_k^0\}_{k\geq 1}$, and a circular order array for the couples (y_i, s_i). Then H_k is formed by updating m times H_k^0, using the m couples $\{(y_i, s_i)\}_{k-m\leq i\leq k-1}$. At the beginning, $H_2^0 = \delta_1\, I$. Then, for $k \geq 2$, the formula for updating the i-th diagonal element of H_k^0 is the following:

$$(H_{k+1}^0)^{(i)} = \left(\frac{\langle H_k^0 y_k, y_k\rangle}{\langle y_k, s_k\rangle (H_k^0)^{(i)}} + \frac{\langle y_k, e_i\rangle^2}{\langle y_k, s_k\rangle} - \frac{\langle H_k^0 y_k, y_k\rangle (\langle s_k, e_i\rangle / (H_k^0)^{(i)})^2}{\langle y_k, s_k\rangle \langle (H_k^0)^{-1} s_k, s_k\rangle} \right)^{-1}\ , \tag{16}$$

where $\{e_i\}_{1\leq i\leq n}$ is an orthonormal basis for the given scalar product on $\mathbb{R}^n$. It was obtained by scaling a diagonalized version of the BFGS formula and proved to be very effective.

	$m=2$	$m=5$	$m=10$	$m=20$	$m=50$
VA14	> 1.92	> 1.92	> 1.92	> 1.92	> 1.92
M1GC3	2.07	1.60	1.46	1.37	1.23
M1QN2	1.49	1.14	1.07	0.97	0.92
M1QN3	1.00	0.85	0.85	0.76	0.74

Table 2: Comparison between the limited memory, CG and BFGS methods.

Table 2 gathers the numerical results from [15], which were obtained with 8 real-life test-problems having a dimension between 34 and 1559 (the average value of n is 692). A comparison is done between the standard BFGS method implemented by Buckley and LeNir (1983), the CG method of Hestenes and Stiefel (1952) implemented in the code VA14 from the Harwell library and three algorithms from the MODULOPT library (INRIA): M1GC3 is the code of Buckley and LeNir (1983), M1QN2 and M1QN3 use Nocedal's proposal with the scaling (15) and (16) respectively. Table 2 gives the number of function/gradient calls divided by the one required by the standard BFGS algorithm. The number m of updates ranges from 2 to 50. The sign $>$ used for VA14 means that the true number should be larger than the one given because the algorithm failed on one of the test-problems. Observe that the performance of the algorithms increases with m, except for VA14, which is not a variable storage method. Observe also that M1QN2 and M1QN3 are better than the BFGS method as soon as m is greater than $10\ldots 20$ and $2\ldots 5$, respectively. This improvement seems mainly due to the scaling made by the initial matrix in these algorithms, which acts as a dynamic preconditioner. An algorithm similar to M1QN2 has been tested by Liu and Nocedal (1988) who report similar results in their comparison with the method of Buckley and LeNir.

The convergence of M1QN2 on strongly convex functions has been proved in [26]. However, because of the counter-examples of Powell (1984), there is some doubt about the possibility of extending this result for nonconvex functions. To make this remark clear, observe that, with critical line-searches, the limited memory methods can be seen as a generalization of

the CG method of Hestenes-Stiefel or Polak-Ribière. Indeed, let $m = 1$ and $H_k^0 = I$ in Nocedal's proposal and suppose that critical line-searches are done, i.e., $\langle s_{k-1}, g_k \rangle = 0$. Then, $\langle y_{k-1}, d_{k-1} \rangle = -\langle g_{k-1}, d_{k-1} \rangle = |g_{k-1}|^2$ and using formula (14), we obtain:

$$\begin{aligned} d_k &= -H_k\, g_k \\ &= -\bar{U}(I,\, y_{k-1},\, s_{k-1})\, g_k \\ &= -g_k + \frac{\langle y_{k-1}, g_k \rangle}{\langle y_{k-1}, d_{k-1} \rangle} d_{k-1} \\ &= -g_k + \beta_k^{\mathrm{PR}}\, d_{k-1}\,. \end{aligned}$$

As the PR method with critical line-searches does not converge on nonconvex functions, the limited memory method (at least for $m = 1$ and without scaling) does not converge either.

5 The partitioned quasi-Newton method

The algorithms we have seen so far are suitable for any objective function f. The *partitioned quasi-Newton* (PQN) *method* proposed by Griewank and Toint (1982) is dedicated to the minimization of functions having a particular structure.

A function f is said *partially separable* if it can be written as a sum of p *elements functions* f_i:

$$f(x) = \sum_{i=1}^{p} f_i(x)\,, \quad \forall x \in \Omega$$

and if each of the element functions f_i is invariant on some subspace $E_i \subset \mathbb{R}^n$:

$$f_i(x + e) = f_i(x)\,, \quad \forall e \in E_i\,.$$

A typical case is when the subspaces E_i are spanned by some basic vectors of $\mathbb{R}^n$, which means that the f_i's only depend on few of the n variables. Note that the E_i's need not be disjoint, hence the name of *partial* separability.

In the PQN method, the Hessian of f, $\nabla^2 f(x) = \sum_{i=1}^{p} \nabla^2 f_i(x)$, is approximated at iteration k by a matrix of the form

$$B_k = \sum_{i=1}^{p} B_k^i\,.$$

The matrix B_k^i approximates the Hessian of the i-th element function and is updated to satisfy an element-wise secant condition:

$$B_{k+1}^i\, s_k = y_k^i := \nabla f_i(x_{k+1}) - \nabla f_i(x_k)\,.$$

This approach has some weak points:

- If the element functions are nonconvex, their Hessian is not positive definite and $\langle y_k^i, s_k \rangle$ is not necessarily positive; therefore the overall approximation is not necessarily positive definite either and may even be singular although the true Hessian is positive definite.

- The preceding point implies that different update formulae (BFGS and SR1) are used, depending on the element function, which is a lack of homogeneity and an additional complexity of the method.
- It requires more effort from the user who has to supply separately the gradient of the element functions.
- The computation of the step by solving $(\sum_{i=1}^{p} B_k^i)\, d_k = -g_k$ can be expensive.

But it has also important nice features:

- if the dimension of the subspaces E_i is large, the element Hessian are identified very quickly because the B^i's have low rank and therefore the overall approximation is rapidly correct;
- the overall approximation is updated by a consistent high rank update;
- the notion of partial separability is basis independent, in contrast with the notion of sparsity.

Liu and Nocedal (1988) have made some numerical comparison between the PQN approach and M1QN2. They report excellent results for the PQN method when the dimension n_i of the subspaces E_i is small. What is disappointing is that the n_i's have to be *very* small (less than $4 \ldots 10$!), otherwise M1QN2 becomes competitive. The comparison was made on the computing time and not on the number of function/gradient calls, which generally favors the PQN code. This seems due to the iteration time of the PQN method that can be $2 \ldots 5$, sometimes $10 \ldots 15$, times larger than the one of M1QN2.

To conclude this section, let us mention that the PQN method has been implemented by Toint (1983) in the Harwell routine VE08.

6 Final remarks

It should be clear at the end of this review that practical optimization remains an art or, at least, a matter for experts. If there is no miracle recipes, it is, however, possible to outline some rules to follow:

- the CG method remains a efficient algorithm for very large-scale problems, say, when only $5n$ memory locations are available; it is also a method to use for large-scale problems when the cost of a *simulation* (the calculation of the function and its gradient) is very cheap: in this case the calculation cost of the algorithm becomes relevant, which favors the very low cost of a CG iteration;
- if the cost of a simulation is important, it is crucial to have a number of function/gradient calls as small as possible; in this case, it is a good idea to use the codes M1QN2 and M1QN3 (Nocedal's approach), which are available in the library MODULOPT of INRIA;
- if the function to minimize has the partial separability property with element functions depending on a small number of variables, then the PQN algorithm is appropriate.

References

[1] **M. Al-Baali (1985).** Descent property and global convergence of the Fletcher-Reeves methods with inexact line search. IMA Journal of Numerical Analysis 5, 121-124.

[2] **M. Al-Baali, R. Fletcher (1986).** An efficient line search for nonlinear least squares. Journal of Optimization Theory and Applications 48, 359-377.

[3] **A. Bachem, M. Grötschel, B. Korte, eds. (1983).** *Mathematical Programming. The State of the Art*. Springer-Verlag, Berlin.

[4] **D. P. Bertsekas (1982).** *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York.

[5] **D. P. Bertsekas, J. N. Tsitsiklis (1989).** *Parallel and Distributed Computation. Numerical Methods*. Prentice-Hall, Englewood Cliffs.

[6] **A. Buckley, A. LeNir (1983).** QN-like variable storage conjugate gradients. Mathematical Programming 27, 155-175.

[7] **R. S. Dembo, T. Steihaug (1983).** Truncated-Newton algorithms for large-scale unconstrained optimization. Mathematical Programming 26, 190-212.

[8] **J. E. Dennis, J. J. Moré (1977).** Quasi-Newton methods, motivation and theory. SIAM Review 19, 46-89.

[9] **J. E. Dennis, R. B. Schnabel (1983).** *Numerical Methods for Unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs.

[10] **L. C. W. Dixon, E. Spedicato, G. P. Szegö, eds. (1980).** *Nonlinear Optimization. Theory and Algorithms*. Birkhäuser, Boston.

[11] **L. C. W. Dixon, G. P. Szegö, eds. (1975, 1978).** *Towards Global Optimization. Vols. 1 and 2*. North-Holland, Amsterdam.

[12] **S. Dolecki, ed. (1989).** *Optimization*. Proceedings of the fifth French-German Conference on Optimization, Varetz, France, October 3-8, 1988. Lecture Notes in Mathematics 1405. Springer-Verlag, Berlin.

[13] **R. Fletcher (1987).** *Practical methods of optimization*. Second edition. J. Wiley & Sons, New York.

[14] **R. Fletcher, C. M. Reeves (1964).** Function minimization by the conjugate gradients. Computer Journal 7, 149-154.

[15] **J. Ch. Gilbert, C. Lemaréchal (1988).** Some numerical experiments with variable storage quasi-Newton algorithms. Mathematical Programming B 45, 407-435.

[16] **J. Ch. Gilbert, J. Nocedal (1990).** Global convergence properties of conjugate gradient methods for optimization. Research Report 1268. INRIA, BP 105, 78153 Le Chesnay, France.

[17] **P. E. Gill, W. Murray (1974).** Newton-type methods for unconstrained and linearly constrained optimization. Mathematical Programming 7, 311-350.

[18] **P. E. Gill, W. Murray, M. H. Wright (1981).** *Practical Optimization*. Academic Press, New York.

[19] **D. Goldfarb (1980).** Curvilinear path steplength algorithms for minimization which use directions of negative curvature. Mathematical Programming 18, 31-40.

[20] **A. Griewank, Ph. L. Toint (1982).** On the unconstrained optimization of partially separable functions, in: M. J. D. Powell, ed., *Nonlinear Optimization 1981*. Academic Press, London.

[21] **W. A. Gruver, E. Sachs (1980).** *Algorithmic Methods in Optimal Control*. Research Notes in Mathematics 47, Pitman, London.

[22] **M. R. Hestenes, E. Stiefel (1952).** Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49, 409-436.

[23] **A. Iserles, M. J. D. Powell, eds. (1987).** *The State of the Art in Numerical Analysis*. Clarendon Press, Oxford.

[24] **C. Lemaréchal (1981).** "A view of line-searches", in: A. Auslender, W. Oettli, J. Stoer, eds., *Optimization and optimal control*. Lecture Notes in Control and Information Science 30 (Springer Verlag, Heidelberg) pp. 59-78.

[25] **C. Lemaréchal (1989).** *Méthodes numériques d'optimisation*. Collection didactique. INRIA, BP 105, 78153 Le Chesnay, France.

[26] **D. C. Liu, J. Nocedal (1988).** On the limited memory BFGS method for large scale optimization. Mathematical Programming B 45, 503-528.

[27] **O. L. Mangasarian, R. R. Meyer, S. M. Robinson, eds. (1981).** *Nonlinear Programming 4*. Academic Press, New York.

[28] **G. P. McCormick (1977).** A modification of Armijo's step-size rule for negative curvature. Mathematical Programming 13, 111-115.

[29] **G. P. McCormick (1983).** *Nonlinear Programming. Theory, Algorithms and Applications*. J. Wiley & Sons, New York.

[30] **M. Minoux (1983).** *Programmation Mathématique. Théorie et Algorithmes*. Dunod, Paris. (two volumes)

[31] **J. J. Moré (1983).** Recent developments in algorithms and software for trust region methods, in: A. Bachem, M. Grötschel and B. Korte, eds., *Mathematical Programming: the State of the Art*, pp. 258-287. Springer Verlag, Berlin.

[32] **J. J. Moré, D. C. Sorensen (1979).** On the use of directions of negative curvature in a modified Newton method. Mathematical Programming 16, 1-20.

[33] **J. J. Moré, D. J. Thuente (1990).** On line search algorithms with guaranteed sufficient decrease. Mathematics and Computer Science Division Preprint MCS-P153-0590, Argonne National Laboratory, Argonne Il 60439.

[34] **G. L. Nemhauser, A. H. G. Rinnooy Kan, M. J. Todd, eds. (1989).** *Optimization*. Handbooks in Operations Research and Management Science, Vol. 1. North-Holland.

[35] **J. Nocedal (1980).** Updating quasi-Newton matrices with limited storage. Mathematics of Computation 35, 773-782.

[36] **D. P. O'Leary (1983).** A discrete Newton algorithm for minimizing a function of many variables. Mathematical Programming 23, 20-33.

[37] **S. S. Oren, E. Spedicato (1976).** Optimal conditioning of self-scaling variable metric algorithms. Mathematical Programming 10, 70-90.

[38] **J.-P. Penot, ed. (1989).** *New Methods in Optimization and their Industrial Uses*. International Series of Numerical Mathematics, Vol. 87. Birkhäuser Verlag, Basel.

[39] **E. Polak, G. Ribière (1969).** Note sur la convergence de méthodes de directions conjuguées. Revue Française d'Informatique et de Recherche Opérationnelle (RAIRO) 16, 35-43.

[40] **M. J. D. Powell (1976).** Some global convergence properties of a variable metric algorithm for minimization without exact line search, in: R. W. Cottle, C. E. Lemke, eds., *Nonlinear Programming*, SIAM-AMS Proceedings, Vol. IX. SIAM Publications.

[41] **M. J. D. Powell (1977).** Restart procedures for the conjugate gradient method. Mathematical Programming 12, 241-254.

[42] **M. J. D. Powell, ed. (1982).** *Nonlinear Optimization 1981*. Academic Press, London.

[43] **M. J. D. Powell (1984).** Nonconvex minimization calculations and the conjugate gradient method, in: D. F. Griffiths, ed., *Lecture Notes in Mathematics 1066*, pp. 122-141. Springer-Verlag, Berlin.

[44] **M. J. D. Powell (1985).** Convergence properties of algorithms for nonlinear optimization. Report DAMTP 1885/NA1. University of Cambridge, England.

[45] **D. F. Shanno (1978).** Conjugate gradient methods with inexact searches. Mathematics of Operations Research 3, 244-256.

[46] **D. F. Shanno, K. H. Phua (1976).** Algorithm 500, minimization of unconstrained multivariate functions. ACM Transactions on Mathematical Software 2, 87-94.

[47] **Ph. L. Toint (1983).** VE08AD, a routine for partially separable optimization with bounded variables. Harwell Subroutine Library, A. E. R. E. (UK).

[48] **Ph. L. Toint (1986).** A view of nonlinear optimization in a large number of variables. Report 86/16. Département de Mathématiques, Facultés Universitaires ND de la Paix, 5000 Namur, Belgium.

[49] **P. Wolfe (1969).** Convergence conditions for ascent methods. SIAM Review 11, 226-235.

[50] **P. Wolfe (1971).** Convergence conditions for ascent methods II: some corrections. SIAM Review 13, 185-188.

[51] **G. Zoutendijk (1970).** "Nonlinear programming, computational methods", in: J. Abadie, ed., *Integer and Nonlinear Programming*, pp. 37-86. North-Holland, Amsterdam.