Survey on nonlinear optimization

Hoon Hong Daniela Văsaru

February 13, 1996

Abstract

In this survey paper, an overview on the different approaches for solving nonlinear optimization problems is given. The presentation includes theory, numeric, interval and symbolic methods.

1 Introduction

Optimization is something that one has to deal with in everyday life. It has applications in any field of science. Just to name a few: management, control theory, physics, engineering.

The goal of this paper is to give an overview on nonlinear optimization both on the theory as on practical methods, putting an emphasis on algorithms. There are already several textbooks and surveys about this field, like [1], [7], [10], [23], [38], [3], [4]. All of them contain some theory and numerical methods. What is new here is the attempt to present all classes of methods used for solving optimization problems: numeric, interval or symbolic. This survey does not pretend to give a full and complete presentation of all optimization algorithms, for this one should consult the references. The main idea here is to give the flavor of what can be done by presenting the main algorithms from the above classes of methods.

A formal definition of the problem to be studied and some more about the different types of optimization problems is given in §2. §3 contains a rough overview of the theoretical aspects: optimality conditions, dual problems. The following sections deal with practical methods for solving optimization problems: numerical methods are presented in §4, interval methods in §5 and symbolic methods in §6.

2 Problem

Let $n \in \mathbb{N}^*$, $m, p \in \mathbb{N}$. Let $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$, be functions and the set M be a nonempty subset of $\mathbb{R}^n$. Indexed variables g_i, h_i will denote the components of the functions g, h.

A general optimization problem has the form:

$$(P) \qquad \begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & g(x) \leq 0, \\ & h(x) = 0, \\ & x \in M. \end{array}$$

The function f is called *objective function*, h, g are the *equality*, respectively *inequality constraints*. There are several types of optimization problems: *discrete* if the functions are over integers, *continuous* if they are defined on the reals; *linear* if all the functions are linear, *nonlinear* otherwise; *unconstrained* if there are no constraints, *constrained* otherwise; *local* when a local minimum of the objective function is sufficient, *global* if the global minimum of it is needed.

This paper focuses on nonlinear continuous optimization.

3 Mathematical foundations

This section contains definitions, notations and some mathematical background related to optimization. A good textbook where one can find more details on this subject is [1].

Assumptions For simplicity of the presentation, unless other stated, assume that the functions f, g, h are twice continuously differentiable.

Notations:

- X subset of $\mathbb{R}^n$;
- $u_0 \in \mathbb{R}$, $u, u^* \in \mathbb{R}^m$, $v, v^* \in \mathbb{R}^p$;
- the variables x, x^* range over the points in $\mathbb{R}^n$;
- d range over the vectors in $\mathbb{R}^n$;
- $\epsilon, \delta, \lambda$ denote positive real numbers;

- $\nabla \bar{f}$ denotes the gradient of a function $\bar{f}$ and $\nabla^2 \bar{f}$ its Hessian; sometimes the Hessian of the objective function of an optimization problem (P) will be denoted by H.

Definition 3.1 (Minimum point) Let $f : X \to \mathbb{R}$ and $x^* \in X$. Then x^* is a *global minimum* for f iff $f(x^*) \leq f(x)$ for all $x \in X$. If the inequality holds only in some local neighborhood of x^*, then x^* is a *local minimum* and if the inequality is strict for $x \neq x^*$ then x^* is a *strict minimum.*

Definition 3.2 (Convex function) The function f is said to be *convex* at a point x^* iff

$$f(\lambda\, x\, +\, (1-\lambda)\, x^*) \leq \lambda\, f(x)\, +\, (1-\lambda)\, f(x^*),$$

for all x and for all $\lambda \in [0, 1]$. If the inequality is strict for $x \neq x^*$ then it is strictly convex. A function convex at all the points of its domain is a convex function. If $-f$ is a convex function then f is called *concave.*

Definition 3.3 (Descent direction) The vector d is called a *descent direction* of f at x iff there exists a δ such that $f(x + \lambda d) < f(x)$, for all $\lambda \in (0, \delta)$.

Definition 3.4 (Optimization terminology) Consider the optimization problem (P).

- The *set of feasible solutions* is defined as:

$$S = \{\, x \in M :\ g(x) \leq 0 \text{ and } h(x) = 0\}.$$

- An *optimal point* of the problem (P) is a minimum point of the objective function restricted at the set of feasible solutions ($f : S \to \mathbb{R}$).

- A *feasible direction* at a point $x \in cl(S)$ is a vector $d \neq 0$, such that for some δ, $x + \lambda d \in S$, for all $\lambda < \delta$.

- An *improving (descent) direction* of the function f at a point x is a vector d, such that for some δ, $f(x + \lambda d) < f(x)$, for all $\lambda < \delta$.

- The set of indexes for the *binding inequality constraints* at x:

$$I_x = \{\, i \in \{1, \cdots, m\}\ :\ g_i(x) = 0\}.$$

  If there is no ambiguity concerning the point x, the subscript will be dropped.

- The *Lagrangian function*, $\Phi : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, associated to problem (P) is defined as:

$$\Phi(x, u, v) = f(x) + u^t g(x) + v^t h(x).$$

- (x^*, u^*, v^*) with $x^* \in M$, $u^* \geq 0$ is a *saddle point* for the Lagrangian function iff:

$$\Phi(x^*, u, v) \leq \Phi(x^*, u^*, v^*) \leq \Phi(x, u^*, v^*)$$

  for all $x \in M$ and (u, v) with $u \geq 0$.

3.1 Unconstrained optimization

Theorem 3.1 (Necessary optimality conditions) *If x^* is a local minimum for f, then:*

- x^* *is* a stationary point,*i.e.* $\nabla f(x^*) = 0$ *(first-order condition),*
- $H(x^*)$ *is positive semidefinite (second-order condition).*

Theorem 3.2 (Sufficient optimality condition) *If $\nabla f(x^*) = 0$ and $H(x^*)$ is positive definite, then x^* is a strict local minimum of f.*

Theorem 3.3 (Necessary and sufficient optimality condition) *Let f be convex at x^*. Then x^* is a global minimum iff $\nabla f(x^*) = 0$.*

It is possible to derive higher-order necessary and sufficient conditions for local optimality.

3.2 Constrained optimization

In the constrained case, the role that stationary points have in unconstrained optimization is taken by the so called Fritz-John (FJ) points and Karush-Kuhn-Tucker (KKT) points. Under certain conditions that the constraints have to satisfy (*constraint qualifications*), a FJ point is also a KKT point. In all the theorems given in this section, M is assumed to be an open set in $\mathbb{R}^n$.

3.2.1 Fritz-John optimality conditions

This conditions were given for the first time by Fritz John ([17]) for the optimization problems with inequality constraints and it was further generalized by Mangasarian ([20]) in order to include also equality constraints.

Definition 3.5 (Fritz-John point) x^* is called a *Fritz-John (FJ) point* (solution) of problem (P) iff there exist $(u_0, u, v) \in \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^p$ (*Lagrange multipliers*) such that:

$$x^* \in S \qquad (PF)$$

$$\left.\begin{array}{c} u_0 \nabla f(x^*) \,+\, \nabla g(x^*)^t\, u \,+\, \nabla h(x^*)^t\, v = 0, \\[2ex] u_0 \geq 0,\; u \geq 0, \\[2ex] (u_0, u, v) \neq (0, 0, 0), \end{array}\right\} \qquad (DF)$$

$$u^t\, g(x^*) = 0. \qquad (CS)$$

(PF) (*primal feasibility*), (DF) (*dual feasibility*), and (CS) (*complementary slackness*) are also called the *Fritz John conditions.* One can see from the above definition that the Lagrange multipliers corresponding to non-binding inequality constraints are 0 and the one corresponding to equality constraints are not required to be positive.

The relations between FJ points and optimal solutions of the problem (P) are given by the following theorems:

Theorem 3.4 (FJ necessary conditions) *If x^* is a local optimal solution of problem (P), then x^* is a FJ-point.*

Theorem 3.5 (FJ sufficient conditions) *Let x^* be a FJ solution of (P). Define $S_1 = \{\, x\, : h(x) = 0 \text{ and } g_i(x) \leq 0, \text{for}\, i \in I\}$. If*

- *h_j are affine and $\nabla h_j(x^*)$ are linearly independent, for $j = 1, \cdots, p$;*
- *there exists an ϵ-neighborhood N_ϵ of x^* such that f and g_i, for $i \in I$ are convex in $S_1 \cap N_\epsilon$,*

then x^* is a local optimal solution for (P).

The problem with FJ-conditions is that for $u_0 = 0$ they merely state that the gradients of the binding inequality constraints at x^* and the ones of the equality constraints are linearly dependent. This is true whenever one has a redundant constraint so it does not provide much information. By imposing some supplementary conditions (constraint qualifications) on the constraint set one can guarantee that at any local minimum there exist Lagrange multipliers with $u_0 > 0$.

3.2.2 Karush-Kuhn-Tucker Optimality conditions

The KKT conditions were developed independently by Karush ([18]) and by Kuhn and Tucker ([19]). Assume the gradients of the binding inequality constraints $\nabla g_i(x^*)$, for $i \in I$ and of the equality constraints $\nabla h_i(x^*)$, for $i = 1, \cdots p$ are linearly independent (*regular* constraint qualification).

Definition 3.6 (KKT point) x^* is called a *Karush-Kuhn-Tucker (KKT) point* (solution) of problem (P) iff there exist $(u, v) \in \mathbb{R}^m \times \mathbb{R}^p$ (*Lagrange multipliers*) such that:

$$x^* \in S \qquad (PF')$$

$$\left.\begin{aligned} \nabla f(x^*) + \nabla g(x^*)^t\, u + \nabla h(x^*)^t\, v = 0, \\ u \geq 0, \end{aligned}\right\} \quad (DF')$$

$$u^t\, g(x^*) = 0. \qquad (CS')$$

(PF'), (DF'), (CS') represent the KKT conditions (primal and dual feasibility, complementary slackness).

Theorem 3.6 (KKT necessary optimality condition) *Let x^* be a local optimal solution for (P). Then x^* is a KKT point.*

Under suitable convexity assumptions, KKT conditions are sufficient for optimality.

Theorem 3.7 (KKT sufficient optimality condition) *Let x^* be a KKT point with Lagrange multipliers u, v. Let $J = \{i : v_i > 0\}$ and $K = \{i : v_i < 0\}$. If the following conditions hold:*

- *f and g_i, for $i \in I$, are convex;*
- *h_i is convex for $i \in J$ and concave for $i \in K$,*

then x^* is a global optimal solution of problem (P). If the above convexity assumptions hold only in a neighborhood of x^* then x^* is a local optimal solution.

3.3 Dual problems

Another way to solve an optimization problem is to deal with the associated problem, called *dual problem.*

Definition 3.7 (Primal/Dual problem) Let (P) be an optimization problem. This is also called a *primal problem.*
The associated problem to it, called *Lagrangian dual problem* , is the following:

$$(D) \qquad \begin{array}{ll} \textit{maximize} & \theta(u,v) \\ \textit{subject to} & u \geq 0, \end{array}$$

where $\theta(u,v) = inf\,\{f(x) \,+\, u^t\, g(x) \,+\, v^t\, h(x) : x \in M\}$.

The following theorems and corollaries gives relations between the optimal solutions of the two problems.

Theorem 3.8 (Weak Duality) *Let x be a feasible solution to problem (P) and (u,v) a feasible solution for the problem (D). Then $f(x) \geq \theta(u,v)$.*

Corollary 3.1

- $inf\{f(x) : x \in M,\, g(x) \leq 0,\, h(x) = 0\} \;\geq\; sup\{\theta(u,v) : u \geq 0\}$;
- If $f(x^*) = \theta(u^*,v^*)$, with $u^* \geq 0$ and $x^* \in S$, then x^* and (u^*,v^*) are optimal solutions of the primal, respectively dual problem;
- If $inf\{f(x) : x \in M,\, g(x) \leq 0,\, h(x) = 0\} = -\infty$ then $\theta(u,v) = -\infty$ for each $u \geq 0$;
- If $sup\{\theta(u,v) : u \geq 0\} = \infty$ then the primal problem has no feasible solutions.

In other words, what one gets from the weak duality theorem is that the optimal value of the objective function in the primal problem is always greater or equal than the optimal value of the objective function of the dual problem. In case the inequality is strict, then a *duality gap* is said to exist.

Theorem 3.9 (Strong Duality) *Let M be convex, f, g convex functions and h affine. Assume that $0 \in h(M)$, where $h(M) = \{h(x) : x \in M\}$ and there exist $x^* \in M$ such that $g(x^*) < 0$ and $h(x^*) = 0$. Then,*

$$inf\{f(x) : x \in M, \, g(x) \leq 0, \, h(x) = 0\} \; = \; sup\{\theta(u, v) : u \geq 0\}.$$

Furthermore, if the infimum is finite, then $sup\{\theta(u, v) : u \geq 0\}$ is achieved at (u^*, v^*), with $u^* \geq 0$. If the infimum is achieved at x^* then ${u^*}^t g(x^*) = 0$.

Theorem 3.10 (Saddle-points and duality gap) *The triple (x^*, u^*, v^*) with $x^* \in M$, $u^* \geq 0$ is a saddle point for the Lagrangian function of the problem (P) iff:*

- $\Phi(x^*, u^*, v^*) = minimum \; \{\Phi(x, u^*, v^*) : x \in M\}$;
- $x^* \in S$;
- ${u^*}^t g(x^*) = 0$.

Moreover, (x^*, u^*, v^*) is a saddle point for the Lagrangian function iff x^* and (u^*, v^*) are optimal solutions to the primal, respectively dual problem, with no duality gap.

4 Numerical methods

Most of the numeric methods for solving constrained optimization problems consist in solving a sequence of unconstrained problems derived from the original one. Furthermore, a multidimensional minimization problem is reduced to a sequence of univariate minimizations (sometimes called line minimizations or line searches).

Good textbooks on numerical methods are: [1], [7], [10]. For surveys, see [4], [3]. Algorithms and implementations for the unconstrained case, can be found in [26].

4.1 Univariate minimization

Let $a, b \in \mathbb{R}$. Suppose that the function $f : \mathbb{R} \to \mathbb{R}$ has a minimum point in $[a, b]$. The interval $[a, b]$ is called *interval of uncertainty*. What one wants is to make this interval as small as possible, say of length l, in order to find a good approximation of the minimum. In other words:

GIVEN: $f : [a, b] \to \mathbb{R}$,

FIND: $x^* \in [a, b]$ an approximate local minimum of f.

There are two classes of methods solving this problem: one that uses just function evaluations and one that uses also derivative information. The first class of methods are based on a property of strictly convex functions given by the following theorem.

Theorem 4.1 *Let function f be strictly convex over the interval $[a, b]$. Let $c, \hat{c} \in [a, b]$ with $c < \hat{c}$.*

- *If $f(c) > f(\hat{c})$ then $f(x) \geq f(\hat{c})$, for all $x \in [a, c)$.*
- *If $f(c) \leq f(\hat{c})$ then $f(x) \geq f(c)$, for all $x \in (\hat{c}, b]$.*

As the theorem says, it is enough to evaluate a function in two points inside the interval of uncertainty in order to be able to reduce it. By applying iteratively this process one can get an interval of desired size. See figure 1 for an algorithm.

A question that arises naturally is how to choose the points $c_k, \hat{c}_k$.

4.1.1 Dichotomous search

One strategy for choosing the evaluation points is to take them in such a way that at each step the maximum of the two possible resulting intervals has minimum size. This can be obtained when both points are in the middle. As they have to be different, a good compromise is to take them symmetrically, at some small distance ϵ from the midpoint of the uncertainty interval:

$$c_k = \frac{a_k + b_k}{2} - \epsilon, \;\; \hat{c}_k = \frac{a_k + b_k}{2} + \epsilon.$$

The number of function evaluations for reducing the uncertainty interval to length l is given by the smallest positive integer n satisfying the inequality:

$$I := \text{Line-Search } ([a_1, b_1], f, l)$$

Input: $f : [a_1, b_1] \to \mathbb{R}$ strictly convex;
l desired length of the uncertainty interval;

Output: I desired uncertainty interval.

(1) [Initialize] $k := 1$;

(2) [Loop]
while $b_k - a_k < l$ do

- compute $c_k, \hat{c}_k$;
- if $f(c) \leq f(\hat{c})$ then $a_{k+1} := a_k$; $b_{k+1} := \hat{c}_k$;
  else $a_{k+1} := c_k$; $b_{k+1} := b_k$;
- $k := k + 1$;

(3) [Return] $[a_k, b_k]$.

Figure 1: General algorithm searching minimum of strictly convex functions.

$$\left(\frac{1}{2}\right)^{n/2} \leq \frac{l}{b-a}.$$

4.1.2 Reusing information

Another strategy is to choose the points such that at each iteration step some information from the previous step is reused: one of the two points will be taken as evaluation point in the next iteration step. In this way the number of function evaluations is just one pro iteration instead of two as it was for the previous method. The methods applying this strategy (*golden-search*, *Fibonacci search*), reveal to be more efficient.

The points are chosen at each step k such that the length of the outcoming interval is independent of the function value, i.e. $\hat{c}_k - a_k = b_k - c_k$. This implies that the two points are related in the following way:

$$\begin{aligned} c_k &= a_k + \alpha_k \,(b_k - a_k), \\ \hat{c}_k &= a_k + (1 - \alpha_k)\,(b_k - a_k), \end{aligned}$$

where $\alpha_k \in \left(0, \frac{1}{2}\right)$. Furthermore, if one wants to preserve one of the points for the next iteration, i.e. one will have either $c_{k+1} = \hat{c}_k$, or $\hat{c}_{k+1} = c_k$, then α_k has to satisfy the following recurrence relation:

$$(1 - \alpha_k)(1 - \alpha_{k+1}) = \alpha_k, \text{for all } k \geq 1. \tag{1}$$

Different choices of α_k lead to different methods.

4.1.3 Golden search

One natural choice is to take α_k constant. The solution of the equation (1) holds $1 - \alpha \approx 0.618$, *the golden number*, from where the name of the method is derived.

The number of function evaluations needed to reduce the length of the uncertainty interval to l is given by the smallest n satisfying:

$$(0.618)^{n-1} \leq \frac{l}{b-a}.$$

4.1.4 Fibonacci search

This method is based on the Fibonacci sequence $\{F_i\}$. Let n be the number of planned function evaluations. Then α_k is changed at every step and is given by:

$$\alpha_k = \frac{F_{n-k-1}}{F_{n-k+1}}, \text{ for } k = 1, \cdots, n-1.$$

It is straightforward to see that α_k verifies equation (1). From the definition of the Fibonacci sequence it comes out that the expressions for the evaluation points are:

$$c_k = a_k + \frac{F_{n-k-1}}{F_{n-k+1}}(b_k - a_k),$$

$$\hat{c}_k = a_k + \frac{F_{n-k}}{F_{n-k+1}}(b_k - a_k),$$

for all $k = 1, \cdots, n-1$.

The number of function evaluations, which has to be known in advance, is given by the smallest n satisfying:

$$F_n \geq \frac{b-a}{2}.$$

As one can see from the expressions for the number of function evaluations, the most efficient method is Fibonacci search, followed by the golden search and at the end the dichotomous search. However, the asymptotical behavior of the Fibonacci search and golden search methods is the same.

4.1.5 Using derivatives

Most frequently, the function to be minimized is not strictly convex so one cannot apply the above algorithms. Another approach to solve the problem is to find the stationary points of the function. This implies knowledge about the derivative. The methods used for finding stationary points are basically methods for computing the zero of a function, in this case the derivative. The most used ones are: the bisection method and Newton's method. As the purpose of this paper is not the computation of zeros, the interested reader is refered to any textbook on numerical analysis or zero computation.

4.2 Multivariate minimization

The next thing one needs to know is to minimize a function of several variables. The problem to be solved is:

GIVEN: $f : \mathbb{R}^n \to \mathbb{R}$,

FIND: x^* an approximation of a (local) minimum of f.

As the methods are iterative, they start with some point $x_1 \in \mathbb{R}^n$ and construct a sequence $\{x_k\}$ of approximations to the local minimum until some termination condition is met. There are two main approaches for moving from one point to the other: *line-searches* and *trust-regions*. Another method that does not fall in any of these categories is the *downhill Simplex method*.

Line-search Approach: movement is always done along lines. If x_k is the current point, then one has to choose a *search direction* d_k and then choose the *step size* λ_k on the line determined by x_k and d_k ($x_k + \lambda\, d_k$ for $\lambda \in \mathbb{R}$). Therefore, the new trying point is of the form: $x_{k+1} = x_k + \lambda_k d_k$. The key parts of these methods are:

- **Choosing the step size:** usually the step is taken such that the function f is minimized along the line:

$$\lambda_k := argmin \{f(x_k + \lambda\, d_k) : \lambda \in \mathbb{R}\}.$$

  Finding the step can be done with any univariate minimization method. Another possibility is to take *discrete steps* on the line that do not lead to the minimum of the function but they insure a decrease of it. For the sake of uniformity, all methods presented here are using line searches (minimization).

- **Choosing the search directions:**
  - **orthogonal directions:** For each iteration point, a sequential minimization along n orthogonal directions (e.g. coordinate axes) is done in order to produce the next iteration point: cyclic coordinate method, Hooke and Jeeves method, Rosenbrock method;
  - **descent directions** Every search direction is a descent direction of the function f. At one extreme there is the *steepest-descent method*, with a slow but global convergence. At the opposite extreme is the *Newton method*, very fast (quadratic) but it does not have global convergence: the starting point has to be near to a stationary point.

Trust-region Approach: a quadratic model of the function at the current iterate is minimized, subject to the constraint that the new trying point remains in a local neighborhood of the old one.

4.2.1 Cyclic Coordinate method

An easy choice for the set of search directions is to take the coordinate axes e_i, for $i = 1, \cdots, n$, where e_i is the vector having 1 on the i-th position and zero on the other. At each iteration point, the function is minimized first along e_1, then the result along e_2 and so on, the final result being the new iteration point. Such a step performed along a set of directions $d_1, \cdots, d_n$ is referred as *exploratory step*. In other words for each iteration step a loop of n steps is performed. The algorithm is given in figure 2.

Theorem 4.2 (Convergence) *Let f be differentiable and x_1 be the starting point of the cyclic coordinates algorithm. If the following conditions hold:*

$$x^* := \text{Cyclic-Coordinate} \quad (f, x_1, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, to be minimized,
x_1 starting point,
ϵ termination scalar,

Output: x^* approximate local minimum of f.

(1) [Initialize] $k := 1$;

(2) [Loop]
repeat
- (a) $y_1 := x_k$;
- (b) [Exploratory step]
  for $j = 1$ to n do
  - $t_j := argmin \{f(y_j + t\, e_j) : t \in \mathbb{R}\}$;
  - $y_{j+1} := y_j + t_j\, e_j$;
- (c) $k := k + 1$; $x_k := y_{n+1}$;

until $||x_k - x_{k-1}|| < \epsilon$.

(3) [Return] x_k.

Figure 2: Cyclic coordinate algorithm.

- *the minimum of f along any line in $\mathbb{R}^n$ is unique;*
- *the sequence of points $\{x_k\}$ generated by the algorithm is contained in a compact set of $\mathbb{R}^n$,*

then the algorithm converges to a stationary point of f.

The method can be applied also to nondifferentiable functions but then it might terminate at a non-optimal point. In this cases, cyclic coordinate search can be combined with an alternative search along the line $x_{k+1} - x_k$: Hooke and Jeeves method. The new method can also be applied when the function is differentiable and it results in an acceleration of the convergence. The step is along the line $x_{k+1} - x_k$ is usually referred as *acceleration step* or *pattern-search step*.

$$x^* := \text{Hooke-Jeeves} \quad (f, x_1, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$,
x_1 starting point,
ϵ termination scalar,

Output: x^* approximate local minimum of f.

(1) [Initialize] $k := 1$; $y_1 := x_1$;

(2) [Loop]
for $k = 2$ to ∞ do

  (a) [Exploratory step]
  for $j = 1$ to n do
  - $t_j := argmin \ \{f(y_j + t\, e_j) : t \in \mathbb{R}\}$;
  - $y_{j+1} := y_j + t_j\, e_j$;

  $x_k := y_{n+1}$;

  (b) [Test for termination]
  if $\|x_k - x_{k-1}\| < \epsilon$. then *break*;

  (c) [Acceleration step]
  - $d := x_k - x_{k-1}$;
  - $\bar{t} := argmin \ \{f(x_k + t\, d) : t \in \mathbb{R}\}$;
  - $y_1 := x_k + \bar{t}\, d$;

(3) [Return] x_k.

Figure 3: Hooke and Jeeves algorithm.

4.2.2 Hooke and Jeeves method

This method is very similar to the previous one, the only difference is that, except for the first iteration, each iteration step consists of an acceleration step and an exploratory one, see figure 3. Initially, the method was designed using discrete steps (see [16]).

Theorem 4.3 (Convergence) *Let f, be differentiable and x_1 the starting point of the Hooke and Jeeves algorithm. If the following conditions hold:*

- *the minimum of f along any line in $\mathbb{R}^n$ is unique,*

- *the level set $L = \{x : \ f(x) \leq f(x_1)\}$ is compact,*

then the algorithm converges to a stationary point.

4.2.3 Rosenbrock method

Rosenbrock's method uses a search along n orthonormal, independent directions, where each exploratory step has a new set of directions. The initial set is usually taken as the directions of the coordinate axes e_i, for $i = 1, \cdots, n$. The algorithm is similar to the cyclic coordinate method, with the difference that after each step one has to compute the new set of directions.

Rosenbrock proposed the method using discrete steps ([31]) and later on was modified by Swan ([34]) by introducing line searches.

Let $d_1, \cdots, d_n$ be the set of directions used at iteration k to compute the new trying point x_{k+1}. Let $d = x_{k+1} - x_k$. Then d can be expressed as $d = \sum_{i=1}^{n} \lambda_i \, d_i$, as a consequence of the linear independency and orthonormality of d_i, $i = 1, \cdots, n$. The new directions $\bar{d}_1, \cdots, \bar{d}_n$ are formed by a Graham-Schmidt procedure as follows:

$$a_j = \begin{cases} d_j, & \text{if } \lambda_j = 0 \\ \sum_{i=j}^{n} \lambda_i \, d_i, & \text{if } \lambda_j \neq 0, \end{cases}$$

$$b_j = \begin{cases} a_j, & \text{if } j = 1 \\ a_j - \sum_{i=1}^{j-1} (a_i{}^t \, \bar{d}_i) \, \bar{d}_i, & \text{if } j \geq 2, \end{cases}$$

$$\bar{d}_j = \frac{b_j}{||b_j||}.$$

for $j = 1, \cdots, n$.

The algorithm is convergent under the same assumptions as for the cyclic coordinate method.

4.2.4 Two extremes: Steepest Descent and Newton's method

When the gradient of the function to be minimized is known, a very simple and straightforward idea is to choose the search direction always in the direction of the steepest descent, i.e. in the direction of the negative gradient $-\nabla f$. The method was proposed by Cauchy in the middle of last century. An algorithm is given in figure 4.

$x^* :=$ Steepest-Descent (f, x_1, ϵ)

Input: $f : \mathbb{R}^n \to \mathbb{R}$, differentiable,
x_1 starting point,
ϵ termination scalar,

Output: x^* approximate local minimum of f.

(1) [Initialize] $k := 1$;

(2) [Loop]
while $\|\nabla f(x_k)\| \geq \epsilon$ do
  (a) [Compute direction] $d_k := -\lambda f(x_k)$;
  (b) [Minimize] $\lambda_k := argmin\ \{f(x_k + \lambda\, d_k) : \lambda \geq 0\}$;
  (c) [Compute next point] $x_{k+1} := x_k + \lambda_k\, d_k$; $k := k + 1$;

(3) [Return] x_k.

Figure 4: Steepest-descent algorithm.

Theorem 4.4 (Convergence) *Let f, be continuously differentiable and x_1 the starting point of the steepest descent algorithm. If the sequence $\{x_k\}$ generated by the algorithm is contained in a compact set, then the algorithm converges to a stationary point.*

An advantage of this algorithm is that is simple and it does not require much storage, $O(n)$. However, its disadvantage, the slow convergence (linear), heavily weights on its use. As soon as it approaches a stationary point the algorithm starts zigzagging, the convergence decreases.

The other easy choice for finding a minimum point is Newton's method for solving systems of equations. If one applies Newton's method for finding a stationary point of the function f, the recurrence relation between the iteration points is:

$$x_{k+1} = x_k - H(x_k)^{-1} \nabla f(x_k). \tag{2}$$

One could interpret this formula as a step of length one in the direction of a deflected steepest descent, where the deflection factor is $H(x_k)^{-1}$. This idea of deflecting the gradient was used later on in other methods.

Newton's algorithm converges quadratically but it does not have a global convergency, the starting point can influence it. Also it requires a $O(n^2)$

$$x^* := \text{Conjugate-Directions}(f, x_1, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, differentiable,
x_1 starting point,
ϵ termination scalar,

Output: x^* approximate local minimum of f.

(1) [Initialize] $k := 1; j := 1; y_1 := x_1$;
(2) [Loop]
while $\|\nabla f(y_j)\| \geq \epsilon$ do
  (a) [Compute search direction] d_j;
  (b) [Minimize] $\lambda_j := argmin\ \{f(y_j + \lambda\, d_j) : \lambda \geq 0\}$;
  (c) [Compute next point] $y_{j+1} := y_j + \lambda_j\, d_j$;
  (d) [Test for restarting]
    if $j = n$ then
    - $k := k + 1$;
    - $x_k := y_{n+1}$;
    - $j := 1;\ y_1 := y_{n+1}$;

    else $j := j + 1$;
(3) [Return] x_k.

Figure 5: Conjugate-directions algorithm.

memory (for the inverse of the Hessian).

4.2.5 Conjugate directions methods

To overcome the disadvantages of the above methods: cycling through sets of directions, slow or not global convergency, a new class of methods was developed: the one using conjugate directions.

Definition 4.1 (Conjugate directions) Let A be a symmetric matrix. The vectors $d_1, \cdots, d_n$ are called $A - conjugate$, if they are linearly independent and $d_i{}^t\, A\, d_j = 0$ for $i \neq j$.

An important property of the conjugate directions is that for quadratic functions, the optimal point is found after n line minimizations along H-conjugate vectors. A general algorithm for this kind of methods is given in figure 5.

The search directions generated in the algorithm are descent directions and if the function is quadratic they are also H-conjugate.

There are two types of methods following this scheme: quasi-Newton methods and conjugate gradient methods.

Quasi-Newton methods (variable metric) The search directions are computed according to the following rule:

$$d_j = -D_j \times \nabla f(y_j),$$

for $j \in \mathbb{N}^*$, where D_j is a symmetric, positive definite, n-square matrix. Usually D_1 is taken the unit matrix. The methods use the same approach as Newton's method for deflecting the gradient, i.e. the deflection factor is a positive definite matrix. Thus, the name that they have.

Examples of variable-metric methods are:

- **Davidon-Fletcher-Powell** (DFP) method ([5], [8]) - the matrix D_j is given by:

  $$D_{j+1} = D_j + \frac{s_j {s_j}^t}{{s_j}^t v_j} - \frac{D_j v_j {v_j}^t D_j}{{v_j}^t D_j v_j},$$

  with

  $$s_j = y_{j+1} - y_j,$$
  $$v_j = \nabla f(y_{j+1}) - \nabla f(y_j),$$

  for $j \geq 1$.

- **Broyden-Fletcher-Goldfarb-Shanno** (BFGS) method ([2], [6], [11], [32]) - the updating of the matrix is very similar to the DFP one:

  $$D_{j+1} = D_j + \frac{v_j {v_j}^t}{{v_j}^t s_j} - \frac{D_j s_j {s_j}^t D_j}{{s_j}^t D_j s_j}.$$

In case the function is quadratic, the two methods coincide. Furthermore, the matrix D_{n+1} is exactly the inverse of the Hessian matrix H. In fact, the quasi-Newton methods are building approximations of H^{-1} at each step in the equation (2).

Conjugate-gradient methods(fixed metric) The search directions are obtained by deflecting the gradient using the previous direction. The updating rule is:

$$d_1 = -\nabla f(y_1),$$

$$d_{j+1} = -\nabla f(y_{j+1} + r_j \times d_j, \text{for } j \geq 1,$$

where r_k is a suitable deflection parameter that characterizes a particular conjugate gradient method. One has:

- **Fletcher - Reeves** method ([9]):

$$r_j = \frac{\|\nabla f(x_{j+1})\|^2}{\|\nabla f(x_j)\|^2};$$

- **Polak - Ribiere** method ([25]):

$$r_j = \frac{\|\nabla f(x_{j+1})\|^2 - \langle \nabla f(x_{j+1}), \nabla f(x_j) \rangle}{\|\nabla f(x_k)\|^2};$$

- **Hestenes - Stiefel's** method([15]):

$$r_j = \frac{\nabla f(x_{j+1})^t v_j}{{d_j}^t v_j}.$$

Theorem 4.5 (Convergency of conjugate directions methods) *Let x_1 be the starting point of such an algorithm. If the level set $L = \{x : f(x) \leq f(x_1)\}$ is compact, then the algorithm converges to a stationary point.*

The convergence rate for both types of algorithms is super-linear.

4.2.6 Trust-region methods

Let q_k be the local quadratic model of f at point x_k:

$$q_k(s) = f(x_k) + \nabla f(x_k)^t\, s + \frac{1}{2} s^t\, H_k\, s,$$

where $H_k = \nabla^2 f(x_k)$, $s = x - x_k$. The trying step is computed as:

$$s_k = argmin\{q_k(s) \,:\, \|s\| \leq \Delta_k\},$$

where $\Delta_k > 0$ is the trust region parameter.

$x^* :=$ Trust-region $(f, x_1, \Delta_1, \epsilon)$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, to be minimized,
x_1 starting point,
Δ_1 initial trust-region parameter,
ϵ termination scalar,

Output: x^* an approximate local minimum of f (critical point of f).

(1) [Initialize] $k := 1$;

(2) [Loop]
while $\|\nabla f(x_k)\| > \epsilon$ do
  (a) [Compute step] $s_k = argmin\{q_k(s) \, : \, \|s\| \leq \Delta_k\}$;
  (b) [Compute agreement ratio]

$$R_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - q_k(s_k)};$$

  (c) [Adjust region]
  if $R_k < 0.25$ then $\Delta_{k+1} := \Delta_k/4$;
  else if $R_k > 0.75 \wedge s_k = \Delta_k$ then $\Delta_{k+1} := 2\,\Delta_k$;
  else $\Delta_{k+1} := \Delta_k$;
  (d) [Make step] if $R_k < 0$ then $x_{k+1} := x_k$;
  else $x_{k+1} := x_k + s_k$;
  $k := k + 1$;

(3) [Return] x_k.

Figure 6: A trust-region algorithm.

In other words, the local quadratic model is minimized subject to the constraint that the new iteration point will be at the distance at most Δ_k from the previous one. Then the ratio R_k between the predicted and actual reduction in the function value is computed.

Depending on R_k that gives the agreement between the two reduction values, the trust region is increased, reduced or remains the same. Also R_k determines if the step s_k is accepted or not. The summary of the algorithm is given in figure 6.

If the function f is twice continuously-differentiable, the algorithm converges to a critical point x^* of f with positive semidefinite Hessian ($\nabla f(x^*) = 0$, $H(x^*)$ is positive definite).

For more details about the trust-region methods, the reader is refered to More ([21], [7]).

4.2.7 Downhill Simplex method

This method is one that does not require any information about the gradient or the Hessian and it is good to use for non-smooth functions.

A *(regular) simplex* in $\mathbb{R}^n$ is defined by a set of $n+1$ (echidistant) points. The method starts with an initial simplex that encloses the minimum of the function. According to the function values on the points defining the simplex ($P_i, i = 1, \cdots, n+1$), the simplex is modified in an "amoeba-like style". The approach due to Spendley (1962) uses a regular simplex ([33]). In order to make a move it finds the simplex vertex with the highest function value and it reflects it through the opposite face. Then it repeats the process with the new simplex with the modification that the highest value is searched only among the old vertexes in order to avoid oscillations. Also if some vertex remains part of the simplex for several stages then the simplex is contracted toward this vertex. Other approaches, e.g. Nelder and Mead (1964) are using non-regular simplexes ([22]) . The movements are similar but they also introduce the expansion on the direction of the new vertex obtained by reflection if the value of the function on this node is smaller then in all the other nodes (the direction is a good one). In order to improve the convergency, restarting has to be used.

4.3 Constrained minimization

There are mainly three types of numeric methods for solving constrained optimization problems: penalty methods, barrier methods, feasible-directions methods.

The first two classes of methods generate a sequence of points approaching an optimal solution from the outside, respectively inside, of the feasible solutions set S. These points are obtained by solving a sequence of multivariate unconstrained minimization problems.

The other class, of feasible-direction methods, is similar to the line-search methods for multivariate minimization. At each step a feasible, improving direction is generated and the function is optimized along it. Thus, the problem is reduced to a serie of univariate minimization problems.

Other methods that reduce the solving of the initial problem to a sequence of simpler optimization problems are:

- cutting planes method - the optimization problem solved at each iteration has linear constraints, the objective function is the same as the one of the initial problem;
- succesive linear programming approach (SLP) - at each step a linear programming problem is solved, problem obtained by using linear approximations to the objective function and the constraints;
- succesive quadratic programming approach - at each step a quadratic optimization problem is solved where the objective function is a quadratic approximation of the initial one and the constraints are linear approximations to the initial ones.

4.3.1 Penalty methods

This methods reduce the constrained problem (P) to a sequence of unconstrained minimization problems by adding to the objective function a *penalty function*. Intuitively, a penalty function introduces for each violated constraint a penalization term.

Definition 4.2 (Penalty function/problem) A continuous function $\alpha : \mathbb{R}^n \to \mathbb{R}_+$ is called a *penalty function* for the optimization problem P if:

- $\alpha(x) = 0$ for $x \in S$ (x feasible point);
- $\alpha(x) > 0$ for $x \in M \setminus S$ (x infeasible point).

The function $f(x)+\mu\alpha(x)$ is called *auxiliary function*, where μ is the *penalty parameter*. The penalty problem associated to (P) is:

$$x^* := \text{Penalty}\ (f, \alpha, x_1, \mu_1, \beta, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, continuous,
$\alpha : \mathbb{R}^n \to \mathbb{R}$ penalty function,
x_1 starting point,
μ_1 initial penalty parameter,
$\beta > 1$ increasing rate of the penalty parameter,
ϵ termination scalar,

Output: x^* approximate optimal local solution of (P).

(1) [Initialize] $k := 1$;
(2) [Loop]
repeat
(a) [Solve unconstrained problem] using x_k as starting point and one of the methods for multivariate minimization, compute:

$$x_{k+1} = argmin\{f(x) + \mu_k\, \alpha(x) :\ x \in M\};$$

(b) [Prepare for next iteration] $\mu_{k+1} := \beta\, \mu_k$; $k := k + 1$;
until $\mu_{k-1}\, \alpha(x_k) < \epsilon$.
(3) Return x_k.

Figure 7: General algorithm for penalty methods.

$$\begin{array}{ll} \textit{maximize} & \theta(\mu) \\ \textit{subject to} & \mu \geq 0, \end{array}$$

where $\theta(\mu) = inf\{f(x) + \mu\alpha(x) : x \in M\}$.

Different choices for the penalty function give different methods. The most used penalty functions are of the form:

$$\alpha(x) = \sum_{i=1}^{m} [maximum\{0, g_i(x)\}]^s + \sum_{i=1}^{p} |h_i(x)|^s,$$

where $s \in \mathbb{N}^*$. A general algorithm is given in figure 7. At each step the penalty parameter is increased and an unconstrained problem is solved, where the objective function is the corresponding auxiliary function.

Theorem 4.6 (Convergence) *Let M be a closed set. Assume $S \neq \emptyset$ and each of the unconstrained minimization problems generated by the penalty* *function algorithm has an optimal solution x_k. If the sequence $\{x_k\}$ is contained in a compact subset of M, then the algorithm converges to an optimal solution of the problem (P).*

The points x_k generated by the algorithm are in general infeasible but they approach in limit an optimal solution of the problem (P). For this reason the method is also called *exterior penalty method.*

In practice, it might happen that the objective function is not defined outside the set of feasible solutions. In this case the penalty methods cannot be applied.

4.3.2 Barrier methods

Barrier methods can be used for solving the following type of optimization problems:

$$(P') \qquad \begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & g(x) \leq 0, \\ & x \in M. \end{array}$$

In the case that equality constraints are present, they have to be included in the set M. The reason is that the algorithms assume that the set $G = \{x : g(x) < 0\}$ is nonempty.

As for penalty methods, the optimal point of (P') is found by solving a sequence of unconstrained optimization problems. The auxiliary function is constructed using a barrier function that does not allow the method to step outside the feasible solutions set. In other words it acts like a barrier on the boundary of S. This means that the optimal solution is approached from within the set of feasible solutions and for this reason the methods are sometimes called *interior penalty methods.* They can be used especially in the cases when the functions involved are not defined outside the feasible solutions set.

For a survey on barrier methods, see [38].

Definition 4.3 (Barrier function/problem) A function $B : \mathbb{R}^n \to \mathbb{R}_+$ is called a barrier function for problem (P') if:

$$B(x) = \sum_{i=1}^{m} \beta(g_i(x)),$$

$$x^* := \text{Barrier}\ (f, B, x_1, \mu_1, \beta, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, continuous,
$B : \mathbb{R}^n \to \mathbb{R}$ barrier function,
x_1 starting point,
μ_1 initial barrier parameter,
$\beta \in (0, 1)$ decreasing rate of the barrier parameter,
ϵ termination scalar,

Output: x^* approximate local optimal solution of (P').

(1) [Initialize] $k := 1$;
(2) [Loop]
repeat
(a) [Solve as unconstrained problem] using x_k as starting point and one of the methods for multivariate minimization, compute:

$$x_{k+1} = argmin\{f(x) + \mu_k\, B(x) :\ x \in M,\ g(x) < 0\};$$

(b) [Prepare for next iteration] $\mu_{k+1} := \beta\, \mu_k$; $k := k + 1$;
until $\mu_{k-1}\, B(x_k) < \epsilon$.
(3) Return x_k.

Figure 8: General algorithm for barrier methods.

where $\beta : \mathbb{R} \to \mathbb{R}$ is continuous over $\{y : y < 0\}$ and satisfies:

$$\beta(y) \geq 0, \text{ if } y < 0 \ \text{ and } \lim_{y \to 0^-} \beta(y) = \infty.$$

The function $f(x) + \mu B(x)$ is an auxiliary function of (P'). The barrier problem associated to (P') is:
minimize $\theta(\mu)$
subject to $\mu \geq 0$,
where $\theta(\mu) = inf\{f(x) + \mu B(x) : g(x) < 0,\ x \in M\}$.

Examples of used barrier functions are:

$$B(x) = \sum_{i=1}^{m} \frac{1}{(-g_i(x))^s},$$

where $s > 0$, or

$$B(x) = -\sum_{i=1}^{m} ln[min\{1, -g_i(x)\}].$$

The general algorithm is given in figure 8.It is similar with the one for penalty methods but here the barrier parameter is decreased in order to obtain convergence.

Theorem 4.7 (Convergence) *Let M be a closed set. Suppose $G \neq \emptyset$ and that the primal optimization problem (P') has an optimal solution x^*. If for any ϵ-neighborhood N of x^* there exist a point $\bar{x} \in M \cap N$ such that $g(\bar{x}) < 0$, then the barrier function algorithm (figure 8) converges to an local optimal point of (P').*

As the barrier function approaches ∞ at the boundary of G,the constraint $g(x) < 0$ can be ignored and the problem to be solved at step (2.a) of the algorithm can be considered as an unconstrained problem, provided that the method used for solving it ensures that the resulting optimal point $x_{k+1} \in G$.

4.3.3 Zoutendijk method of feasible directions

The original algorithm given by Zoutendijk ([39]) does not converge always. The algorithm given here, figure 9, is a modification of the Zoutendijk method due to Topkis and Veinott ([35]) that guarantees its convergence. It can be applied to optimization problems of the form (P'), where $M = \mathbb{R}^n$ (contain only inequality constraints).

Theorem 4.8 (Convergence) *Let f, g be continuously differentiable and $\{x_k\}$ the sequence of points generated by the Zoutendijk - Topkis - Veinott algorithm. Then any accumulation point of $\{x_k\}$ is a FJ-point for the optimization problem considered.*

4.3.4 The Gradient projection method of Rosen

Ity is known that the steepest descent direction of the function f is the one of the negative gradient. In the constrained case, this direction might lead to infeasible points. However, this can be avoided by projecting the gradient and then applying some correction step that leads to the feasible region.

$x^* :=$ Zoutendijk-Topkis-Veinott (f, g, x_1)

Input: $f : \mathbb{R}^n \to \mathbb{R}$, , objective function,
$g : \mathbb{R}^n \to \mathbb{R}^m$, inequality constraints,
x_1 starting feasible point,

Output: x^* FJ-point of (P').

(1) [Loop]
for $k = 1$ to ∞ do
(a) [Compute search direction] let (z_k, d_k) be the optimal solution of the linear optimization problem:

$$\begin{array}{lll} \textit{minimize} & z & \\ \textit{subject to} & \nabla f(x_k)^t d - z \leq 0 & \\ & \nabla g_i(x_k)^t d - z \leq -g_i(x_k), & \text{for } i = 1, \cdots, m \\ & -1 \leq d_j \leq 1 & \text{for } i = 1, \cdots, n \end{array}$$

(b) [Test for termination]
if $z_k = 0$ then break;
(c) [Compute step](here $z_k < 0$) $x_{k+1} = x_k + \lambda_k d_k$, where:

$$\lambda_k = argmin\{f(x_k + \lambda d_k) : 0 \leq \lambda \leq \lambda_{max}\},$$

and $\lambda_{max} = sup\{\lambda : g_i(x_k + \lambda d_k) \leq 0, \text{for } i = 1, \cdots, m\}$.

(3) Return x_k.

Figure 9: Zoutendijk's algorithm with the improvements of Topkis-Veinott.

$$x^* := \text{Rosen} \quad (f, g, h, x_1, \epsilon)$$

Input: $f : \mathbb{R}^n \to \mathbb{R}$, differentiable,
$g : \mathbb{R}^n \to \mathbb{R}^m$, inequality constraints,
$h : \mathbb{R}^n \to \mathbb{R}^p$, equality constraints,
x_1 starting feasible point,

Output: x^* KKT point of (P).

(1) [Loop]
for $k = 1$ to ∞ do

(a) form the matrix Q that has as rows $\nabla g_i(x_k)^t$ for $i \in I$ (the binding inequality constraints at x_k), and $\nabla h_i(x_k)^t$, for $i = 1, \cdots, p$;
(b) [Compute projection matrix] $P = I - Q^t(QQ^t)^{-1}Q$;
(c) $d_k := -P\nabla f(x_k)$;
(d) if $d_k \neq 0$ then goto step(i);
(e) compute $(u^t, v^t) = -\nabla f(x_k)Q^t(QQ^t)^{-1}$;
(f) if $u \leq 0$ then goto step (2) [stop];
(g) update Q by deleting the row corresponding to some negative u_j;
(h) goto step (b);
(i) [get next point]
$\lambda' = argmin \, \{f(x_k + \lambda \, d_k) : \lambda \geq 0\}$;
$x' = x_k + \lambda_k \, d_k$;
apply a correction to x' in order to get a feasible point x_{k+1};

(2) Return x_k.

Figure 10: Rosen's gradient projection algorithm.

Definition 4.4 (Projecting matrix) *A square $n \times n$ matrix P is called a projection matrix if $P^t = P$ and $P\,P = P$.*

Rosen (1960, 1961) ([29], [30]) proposed a projection on the gradient that improves the objective function while maintaining feasibility. An informal algorithm for the nonlinear case, with $M = \mathbb{R}^n$ is given in figure 10 .

The algorithm in its original form does not converge always to a KKT point but it can be modified so that the convergence is guaranteed.

5 Interval methods

One advantage of interval arithmetic is that it can be used in order to get global results: all the zeros of a function, the global maximum of a function and the set of global minimizers. Due to this property the optimization problem that will be treated here is the one of global optimization.

Let $[x]$ be a box in $\mathbb{R}^n$ (an n dimensional interval vector).

GIVEN:	$f : [x] \to \mathbb{R}$ twice continuously differentiable,
FIND:	$\{x^*\}$ the set of all global minimizers of f inside $[x]$,
	f^* the minimum value of the function.

The method that will be given is due to Hansen ([12], [13]), with the modifications of Ratz ([27]). A presentation and implementation of it can be found in [28].

Notations Let $[y] = [\underline{y}, \overline{y}]$ be a box (interval) in $\mathbb{R}^n$. Then

- $m([y])$ its midpoint,
- $d([y])$ its diameter (length),
- $d_{rel}([y])$ the relative diameter,
- $d_{rel,\infty}([y])$ the maximum relative diameter,
- $[f_y] = f_{[]}([y])$ the interval evaluation of function f on $[y]$,
- $f_\diamond$ the floating point interval evaluation of the function f.

Main Ideas:

1. *Subdividing boxes:*

   Starting with the initial box, subdivide it and drop the sub-boxes that do not contain a global minimizer, store the other ones in a list L and repeat the process until the desired accuracy is obtained.

2. *Verification:*

   The last step that remains is to verify the existence and uniqueness of a local minimizer within each sub-box.

Tests used for rejecting sub-boxes:

- **Midpoint test:** determines and improves an upper bound $\tilde{f}$ for f^*. This bound is then used for discarding sub-boxes $[y]$ where the function takes greater values: $\underline{f_y} > \tilde{f}$.

  At the beginning $\tilde{f} = \infty$. Then a box $[z]$ from the list L is taken such that $\underline{f_z} \leq \underline{f_y}$ for all $[y]$ in L. The upper bound $\tilde{f}$ is improved by $\tilde{f} = min\{f(c), \tilde{f}\}$, where $c = m([z])$ (or any other point in $[z]$). Now with this improved value one can discard some more boxes from L.

- **Monotonicity test:** discards sub-boxes where the function is strictly monotone (with some more considerations for boundary cases). It is enough that the function is monotone in a single coordinate direction in order to discard a box: $0 \notin [\nabla f]_i$, for some $i = 1, \cdots, n$.

- **Concavity test:** discard sub-boxes where the function is not convex (the Hessian is not positive semidefinite).

- **Interval Newton Gauss-Seidel step:** refine the sub-boxes by applying one step of the extended interval Newton Gauss-Seidel method for finding a zero of the gradient: if no zero has been found then the sub-box can be discarded, otherwise either a refinement is obtained or a splitting in more sub-boxes.

In figure 11, a simplified version of the algorithm that computes the candidates for global optimizers is given.

In the algorithm, the list L contains pending sub-boxes that have to be refined and the list L_{res} contains the list of the obtained candidates for global optimizers. The boxes $[y]$ are stored in the lists in non decreasing order of $\underline{f_y}$. Furthermore, it is assumed that the monotonicity and concavity test are available. The procedure $NewtonStep(f, [x], [V], p)$ applies one step of the interval Newton Gauss-Seidel method to the gradient of f and returns p sub-boxes stored in V, bracketing the zeros of it in $[x]$.

There were several technical details left out from the algorithm for the sake of simplicity like:

- as the interval function evaluation usually overestimates the interval for the function value, one can use centered forms to overcome this;

- in order to avoid unnecessary computation, one can store together with the box $[y]$ also the value $\underline{f_y}$;

Global-Optimize $(f, [x], \epsilon; L_{res}, [f^*])$

Input: $f : [x] \to \mathbb{R}$, twice continuously differentiable,
ϵ desired accuracy,

Output: L_{res} the list of candidates for global minimizers.
$[f^*]$ interval bracketing the minimum value of f in $[x]$,

(1) [Compute upper bound for f^*: $\tilde{f}$] $c := m([x])$; $[f_c] := f_\diamond(c)$; $\tilde{f} := \overline{f_c}$;
(2) [Initialize] $[y] := [x]$; $L := \{\}$; $L_{res} := \{\}$; (L list of pending sub-boxes)
(3) [Loop]
repeat

- (a) [Bisect $[y]$ in $[u_1], [u_2]$.]
- (b) [Apply tests on the 2 halves]
  for $i = 1$ to 2 do
  - if *MonotonicityTest*$(f, [u_i])$ then reiterate i;
  - [Function value check] if $\tilde{f} < \underline{f_{u_i}}$ then reiterate i;
  - if *ConcavityTest*$(f, [u_i])$ then reiterate i;
  - *NewtonStep*$(f, [u_i], [V], p)$;
  - [Store boxes in the list L]
    for $j := 1$ to p do
    - if *MonotonicityTest*$(f, [V]_j)$ then reiterate j;
    - if $\tilde{f} \geq \underline{f_{V_j}}$ then store V_j in L.
- (d) $Bisect := false$;
- (e) [Store the candidates in the resulting list]
  while $(L \neq \{\})$ and (not $Bisect$) do
  - [Extract box from L] $[y] := Head(L)$; delete $[y]$ from L;
  - [Midpoint test] $\tilde{f} := min\{\tilde{f}, f(m[y])\}$; delete all boxes $[z]$ from L with: $\underline{f_z} > \tilde{f}$;
  - [Update optimal value] $[f^*] := [\underline{f_y}, \tilde{f}]$;
  - [Check for doneness]
    if $(d_{rel}([f^*]) < \epsilon)$ or $(d_{rel,\infty}([y]) < \epsilon)$ then store $[y]$ in L_{res};
    else $Bisect := true$;

until (not $Bisect$);

(4) [Update optimal value] $[y] := Head(L_{res})$; $[f^*] := [\underline{f_y}, \tilde{f}]$;
(5) [Return] $L_{res}, [f^*]$.

Figure 11: Hansen algorithm with Ratz modifications

- the computation of the gradient and Hessian are done by automatic differentiation.

For a complete algorithm and implementation, see [28].

Verification At the end of the algorithm a list of small boxes that could not be discarded was obtained. What should be done now is to verify that each box contains one and only one global minimizer. As there are no known ways of doing it, the verification step consists only in checking the existence and uniqueness of a local minimizer. This is done by first checking the existence and uniqueness of a stationary point of the function using interval Newton Gauss-Seidel method and then by checking the positive definiteness of the Hessian. The result of the verification procedure is the list of candidates for global minimizers with the information that the above conditions were validated or not.

The method can be further extended to treat the case when global minimizers are on the boundary too.

6 Symbolic Methods

Up to now, by solving an optimization problem it was meant finding a point x^* that satisfies the constraints and for which the value of the objective function is minimized. What if somebody is interested in finding the minimum value f^* of the objective function in the constrained region and only after this the point where it is obtained?

For finding the minimum value, a new variable $y \in \mathbb{R}$ and a new function $f(x) - y$ is introduced. Using this new function and the constraints, a predicate on y is produced, such that the minimum value of y for which the predicate is true is the minimum of the function f on the region defined by the constraints.

Finding predicates with the above property is an active area of research in symbolic computation. One such predicate is that y is some root of a polynomial. An idea that works for univariate function optimization is to compute the discriminant of $f(x) - y$ with respect to x and then one gets a nonzero polynomial in y. Unfortunately, generalizing this approach is not so easy. See [36] for the univariate and bivariate case.

There are no textbooks in symbolic methods for nonlinear optimization, as it is a quite new area of research.

6.1 Weispfenning's method for linear and quadratic optimization

Volker Weispfenning gives in [37] a method to construct such a predicate in y, for parametric linear and quadratic optimization problems. The method consists in two steps:

- reduce the optimization problem to a quantifier elimination (QE) one;
- solve the quantifier elimination problem.

The above procedure can be applied to any optimization problem, the only point where the linearity comes into account is in the solving of the (QE) problem.

Reducing to QE
Let the optimization problem under study to be:

GIVEN:	f(p,x), Parametric objective function. C(p,x), Parametric constraints.
FIND:	Parametric minimum value of f(p,x) under C(p,x).

Consider now the following quantifier elimination problem:

GIVEN:	$(\exists x)C(p,x) \wedge f(p,x) \leq y$,
FIND:	F(p,y), a quantifier-free formula, and equivalent to the input.

It is straightforward to prove that the two problems are equivalent.

Theorem 6.1 *Let v be arbitrary but fixed value of the parameter p. The minimum of $f(v,x)$ under the constraint $C(v,x)$ is the same as the minimum of y satisfying $F(v,y)$.*

Thus, if one has a quantifier elimination method, one can use it for solving an optimization one.

Solving the (QE) problem The algorithms given by Volker Weispfenning are for parametric linear and quadratic optimization problems.

The quantifiers are eliminated one at a time. When the function is linear, the ideas of Loos/Weispfenning can be applied, with some further improvements due to the nature of the problem studied. The main idea is to replace a quantifier by a disjunction over a finite set of "sample" points that can be immediately be extracted from C. The improvement that can be done when solving optimization problems is that some sample points (in the non parametric case) can be dropped because they will not produce a minimum of the function f.

If the function is quadratic, i.e. $f(x) = ax^2 + bX + c$, then one just has to add one more sample point, the critical point: $-b/2a$.

Complexity It is quite interesting to note that the complexity of the Weispfenning's method, very similar to the classical one of Fourier-Motzkin, is only single-exponential.

6.2 Using Hermite's theorem

The method given here is derived from Hermite's theorem for multivariate polynomials, see [14] , with further improvements by Pedersen et al. in [24]. This theorem is used for counting the number of real roots of a system of equations. As one of the authors observed, it can be applied also for optimization problems.

First a short presentation of Hermite's theorem is given and then its application to optimization is described.

6.2.1 Hermite's theorem

Let K be a subfield of $\mathbb{R}^n$, $x \in K$ and $f, f_i \in K[x]$ polynomials, for $i = 1, \cdots, s$. Consider the ideal $I = \langle f_1, \cdots, f_s \rangle$ and assume that the vector space generated by it, $A = K[x]/I$, is finite dimensional (the variety $V(I)$ is finite). The real points in the variety $V(I)$ defined by the ideal I are denoted by $V_R(I)$.

Notations:

- $m_g : A \to A$, $m_g(f) = gf$, where $g \in A$;

- $B : A \times A \to K$, $B(f,g) = Tr(m_f m_g) = Tr(m_{fg})$, where Tr is the trace function;

- $B_h : A \times A \to K$, $B_h(f,g) = B(hf,g) = Tr(m_{hfg})$, for some $h \in \mathbb{R}^n$.

Let Q_h be the quadratic form induced by the bilinear form B_h: $Q_h = B_h(g,g)$.

Theorem 6.2 *Let $V(I)$ be a finite affine variety and $h \in K[x]$. Then,*

$$\sigma(Q_h) = \sharp\{x \in V_R(I) : h(x) > 0\} - \sharp\{x \in V_R(I) : h(x) < 0\},$$

where σ is the signature of the quadratic form.

6.2.2 Solving optimization problems

In order to use the previous theorem, the optimization problem considered has just equality constraints.

GIVEN:	f, $h_i, i = 1, \cdots, p$ polynomials in $K[x]$;
FIND:	f^* the minimum value of the objective function f subject to $h_i = 0, i = 1, \cdots, p$.

Recall that there is a one to one correspondence between optimal points of an optimization problem and the saddle-points of the Lagrangian function Φ (theorem 3.10), where $\Phi : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$, $\Phi(x,v) = f(x) + v^t h(x)$. Furthermore, any saddle point of a function is a critical point of it. Therefore, if (x^*, v^*) is a saddle-point of Φ, then $\nabla\Phi(x^*, v^*) = 0$.

Consider from now on that the polynomials f_i from the previous section are defined by the partial derivatives of Φ:

$$f_i = \begin{cases} \partial\Phi/\partial x_i, & i = 1, \cdots, n, \\ \partial\Phi/\partial v_{i-n} & i = n+1, \cdots, n+p. \end{cases} \qquad (*)$$

Remarks

- If the variety $V(I)$ is infinite, the algorithm given here cannot be applied. One of the cases in which this might happen is when there are infinitely many global optimal points, a degenerate case.

- One could also choose not to introduce the Lagrangian and consider the variety generated by the h_i's. The results will remain the same provided that the corresponding variety is finite. However the version given here is more general as one might have an infinite variety corresponding to the h_i's but a finite one when using the Lagrangian.

Let y be an arbitrary but fixed value in $\mathbb{R}$. Let $g = f - y$. If $V(I)$ is finite, denote the corresponding quadratic form, Q_g, by $Q_{f,y}$. Then, by Hermite's theorem:

$$\sigma(Q_{f,y}) = \sharp\{x \in V_R(I) : f(x) > y\} - \sharp\{x \in V_R(I) : f(x) < y\}. \tag{3}$$

Now, take $a, b \in \mathbb{R}$, with $a < b$. From equation (3), it follows:

$$\begin{aligned} \sigma(Q_{f,a}) - \sigma(Q_{f,b}) &= \sharp\{x \in V_R(I) : a < f(x) \leq b\} + \\ &+ \sharp\{x \in V_R(I) : a \leq f(x) < b\} \end{aligned}$$

Under the assumption that $f(x) \neq a$ and $f(x) \neq b$ for all $x \in V_R(I)$ one has:

$$\sharp\{x \in V_R(I) : a < f(x) < b\} = \frac{1}{2}(\sigma(Q_{f,a}) - \sigma(Q_{f,b})). \tag{4}$$

This formula gives the number of points in the variety for which the value of the objective function is in the open interval (a, b).

Algorithmic Idea: The problem is solved iteratively. Suppose that at iteration k one has an interval $[a_k, b_k]$ that includes the value f^*. Then, one can split (for example bisect it) the interval and retain the subinterval that contains f^*. The subinterval to be kept is given by the leftmost subinterval for which the number computed in formula 4 is not zero.

As the starting interval one can take the whole real axis and the process stops when the interval is small enough to approximate with the desired precision the optimal value.

One problem left is how to check if $f(x) \neq y$ for all the points in the variety $V_R(I)$.

Define:

$$\begin{aligned} N_{\tau\, y} &= \sharp\{x \in V_R(I) : g \,\tau\, 0\}, \\ &= \sharp\{x \in V_R(I) : f \,\tau y\} \end{aligned}$$

where $\tau \in \{<, >, =\}$.

$$I := \text{Hermite} \quad (f, h, \epsilon)$$

Input: f objective polynomial function;
$h_i, i = 1, \cdots, p$ polynomial equality constraints;
ϵ desired precision;

Output: I interval of length ϵ containing the minimum value of f^*.

(1) [Initialize] $a := -\infty$; $b := \infty; \Phi := f + v^t h$;
(2) [Construct the polynomials defining the variety]
for $i = 1$ to n do $f_i = \partial\Phi/\partial x_i$;
for $i = n + 1$ to $n + p$ do $f_i = \partial\Phi/\partial v_{i-n}$;
(3) [Test for infinite variety]
if Q_1 can be computed then $\sigma_1 := \sigma(Q_1)$;
else *error: "Infinite variety".*
(2) [Loop]
while $b - a < \epsilon$ do
- (a) [Split the interval]
  repeat
  - find c in (a, b);
  - $g := f - c$;
  - $N_{=c} := \sigma_1 - \sigma(Q_{g^2})$;

  until $N_{=c} = 0$;
- (b) $k := \frac{1}{2}(\sigma(Q_{f,a}) - \sigma(Q_{f,c}))$;
- (c) [Prune]
  if $k = 0$ then $a := c$;
  else $b := c$;

(3) [Return] $[a, b]$.

Figure 12: An algorithm for computing the minimal value of a function based on Hermite's multivariate theorem.

Consider now Hermite's theorem for $h = 1$, $h = g$, respectively $h = g^2$. One obtains:

$$
\begin{aligned}
\sigma(Q_1) &= \sharp\{x \in V_R(I)\} \\
&= \sharp\{x \in V_R(I) : g < 0\} + \sharp\{x \in V_R(I) : g > 0\} + \\
&\quad + \sharp\{x \in V_R(I) : g = 0\} \\
&= N_{<y} + N_{>y} + N_{=y}, \\
\sigma(Q_g) &= N_{<y} - N_{>y}, \\
\sigma(Q_{g^2}) &= \sharp\{x \in V_R(I) : g^2 > 0\} - \sharp\{x \in V_R(I) : g^2 < 0\} \\
&= \sharp\{x \in V_R(I) : g \neq 0\} \\
&= N_{<y} + N_{>y}.
\end{aligned}
$$

Hence:

$$N_{=y} = \sigma(Q_1) - \sigma(Q_{g^2}).$$

By definition, $N_{=y}$ gives the number of points in the variety $V_R(I)$ where the function f takes the value y and this solves the problem.

The main steps of the algorithm are outlined in figure 12.

Some remarks proving the correctness of the algorithm in figure 12:

- the loop in the spliting step will finish after a finite number of steps because of the finiteness of the variety $V_R(I)$;
- it is straightforward to show that $N_{=-\infty} = N_{=\infty} = 0$.

As at each step a quadratic form has to be computed, one has to have fast methods for doing it.

The algorithm was given for the case when one splits the interval in 2 parts. The same ideas work also when dividing in more parts. However, one has to compute several quadratic forms that will not be reused at the next step. This would affect the performance of the algorithm but it could be useful when a parallel algorithm is used.

References

[1] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms, Second Edition.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons , Inc., 1993.

[2] C.G. Broyden. The convergence of a class of double rank minimization algorithms 2. the new algorithm. *J. Institute of Mathematics and Its Applications*, 6:223–246, 1970.

[3] Thomas F. Coleman. Large-scale numerical optimization: Introduction and overview. Technical Report TR91-1236, Cornell University, Dept. of Computer Science, sept 1991.

[4] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Large-scale non-linear constrained optimization: a current survey. Technical Report TR/PA/94/03, CERFACS, Toulouse, France, 94.

[5] W.C. Davidon. Variable metric methods for optimization. Research Development Report ANL-5990, AEC, 1959.

[6] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

[7] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, second edition, 1987. Intermediate. Nice overall survey.

[8] R. Fletcher and M. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963.

[9] R. Fletcher and C. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

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

[11] D. Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[12] E. Hansen. Global optimization using interval analysis - the multi-dimensional case. *Numerische Mathematik*, 34:247–270, 1980.

[13] E. Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker, New York, 1992.

[14] C. Hermite. Remarques sur la theoreme du sturm. *C.R.Acad. sci.Paris*, 36:52–54, 1853.

[15] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Research National Bureau of Standards*, 49:409–436, 1952.

[16] R. Hooke and T.A. Jeeves. Direct search solution of numerical and statistical problems. *J. ACM*, 8:212–229, 1961.

[17] Fritz John. Extremum problems with inequalities as side conditions. In K.O. Friedrichs, O.E. Neugebauer, and J.J. Stoker, editors, *Courant Anniversary Volume*, Studies and Essays. Wiley-Interscience, 1948.

[18] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Dept. of Mathematics, University of Chicago, 1939.

[19] H.W. Kuhn and A.W. Tucke. Nonlinear programming. In J. Neyman, editor, *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, California, 1951. University of California Press.

[20] O.L. Mangasarian and S. Fromovitz. The fritz-john necessary conditions optimality conditions in the presence of equality and inequality constraints. *J. Mathematical Analysis and Applications*, 11:37–47, 1967.

[21] J.J. More. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Groetschel, and B. Korte, editors, *Mathematical Programming: The state of the art*, pages 258–287. Springer-Verlag, Berlin, 1983.

[22] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1964.

[23] G. L. Nemhauser, A. H. G. Rinnooy Kahn, and M. J. Todd. *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*. North Holland, Amsterdam, 1989.

[24] P. Pedersen, M.-F. Roy, and A. Szpirglas. Counting real zeros in the multivariate case. *Progress Math.*, 109:203–223, 1993.

[25] E. Polak and G. Ribiere. Note sur la convergence de methods de directions conjuges. *Revue Francaise Informat, Recherche Operationelle*, 16:35–43, 1969.

[26] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C. The Art of Scientific Computing.* Cambridge University Press, 1988.

[27] D. Ratz. *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen.* PhD thesis, Universitaet Karlsruhe, 1992.

[28] R.Hammer, M. Hocks, U. Kulisch, and D. Ratz. *Numerical Toolbox for Verified Computing I.* Number 21 in Springer Series in Computational Mathematics. Springer-Verlag, Berlin and Heidelberg and New-York, 1993.

[29] J.B. Rosen. The gradient projection method for nonlinear programming, part i, linear constraints. *SIAM J. Applied Mathematics*, 8:181–217, 1960.

[30] J.B. Rosen. The gradient projection method for nonlinear programming, part ii, nonlinear constraints. *SIAM J. Applied Mathematics*, 9:514–553, 1961.

[31] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Computer Journal*, 3:175–184, 1960.

[32] D.F. Shanno. Conditioning of quasi-newton methods for function minimizations. *Mathematics of Computation*, 24:641–656, 1970.

[33] W. Spendley, G.R. Hext, and F.R. Himsworth. Sequential application of simplex designs of optimization and evolutionary operations. *Technometrics*, 4(441-461), 1962.

[34] W.H. Swann. Report on the development of a new direct search method of optimization. Research Note 64/3, Imperial Chemical Industries Ltd. Central Instr. Res. Lab., 1964.

[35] D.M. Topkis and A.F.Veinott. On the convergence of some feasible direction algorithms for nonlinear programming. *SIAM J. Control*, 5:268–279, 1967.

[36] Alexei Yu. Uteshev and Timofei M. Cherkasov. On the search of the maximum of a polynomial. *J. Symbolic Computation*, 11:1–18, 1995.

[37] Volker Weispfenning. Parametric linear and quadratic optimization by elimination. Technical Report MIP - 9404, Passau University, april 1994.

[38] Margaret H. Wright. Interior methods for constrained optimization. *Acta Numerica*, pages 341–407, 1992.

[39] G. Zoutendijk. *Methods of Feasible Directions*. Elsewier, Amsterdam and D. Van Nostrand, Princeton, N.J., 1960.