Introduction à l'optimisation numérique

Jérôme Malick – jerome.malick@imag.fr

Automne 2006

Ces notes forment un support à la seconde moitié du cours d'Optimisation de l'Ensimag 2A du premier semestre 2006-2007. L'objectif de ce cours est d'amener à la maîtrise des méthodes numériques modernes pour résoudre certains types de problèmes d'optimisation issus de l'industrie et des services. Pour cela, les outils mathématiques indispensables sont introduits, et certaines méthodes seront mises en pratique lors de TPs sur machines. Ce document se découpe naturellement en 2 parties mettant en avant les deux thèmes suivants :

1. optimisation différentiable sans contraintes (§1 à 3),
2. optimisation convexe (non différentiable) et dualité lagrangienne (§4 et 5).

Ces notes sont essentiellement issues du poly des années précédentes et de passages de deux ouvrages de référence :

[BGLS]	J.F. Bonnans, J.Ch. Gilbert, C. Lemaréchal, C. Sagastizábal *"Numerical Optimization: theoretical and practical aspects"*, Springer Verlag 2002
[HULL]	J.-B. Hiriart-Urruty, C. Lemaréchal *"Convex Analysis and Minimization Algorithms"* Springer Verlag, 1993 (Grundlehren der mathematischen Wissenschaften, 305 et 306)

Notez enfin ce poly n'est qu'un support de cours, reprenant schématiquement les idées, les principaux résultats et donnant des compléments et des références précises. Le corps du cours - motivations, exemples, applications, preuves et exercices - sera développé en classe. En particulier, nous insisterons sur certains passages de ce poly tandis que d'autres ne seront qu'effleurés.

1 Differentiable optimization

We will focus in this course on optimization problems without constraints: this permits to introduce the general ideas of numerical optimization methods, and to use them on first (but important) examples. So we aim at solving

$$\min\{f(x)\,, \quad x \in \mathbb{R}^n\}. \tag{1}$$

We suppose here in the first three sections that f is a "regular" function (say, C^2 to fix ideas). Later, Section 5 will be devoted to the case of nondifferentiable convex functions (often produced by Lagrangian duality, studied in Section 4).

1.1 Two introductive examples

We begin with two applications illustrating the need for unconstrainted differentiable optimization in important situations. This subsection can be skipped in a first reading.

Molecular Biology [BGLS §1.2.2] An important problem in biochemistry, for example in pharmacology, is to determine the geometry of a molecule. Various techniques are possible (X-ray crystallography, nuclear magnetic resonance,...). We detail one of these (using optimization) which is very convenient when

- the chemical formula of the molecule is known,
- the molecule is not available, making it impossible to conduct any experiment,
- one has some knowledge of its shape and one wants to *refine* it.

The idea is to compute the positions of the atoms in the space that minimize the associated potential energy. Let N be the number of atoms and call $x_i \in \mathbb{R}^3$ the spatial position of the i^{th} atom. To the vector $X = (x_1, \dots, x_N) \in \mathbb{R}^{3N}$ is associated a potential energy $f(X)$ (the "conformational energy"), which is the sum of several terms. For example:

- Bond length: between two atoms i and j at distance $|\, x_i - x_j \,|$, there is first an energy of the type
$$L_{ij}(x_i, x_j) = \lambda_{ij}(\,|\, x_i - x_j \,| - d_{ij})^2 \,.$$
- There is also a Van der Waals energy, say
$$V_{ij}(x_i, x_j) = v_{ij}\Big(\frac{\delta_{ij}}{|\, x_i - x_j \,|}\Big)^6 - w_{ij}\Big(\frac{\delta_{ij}}{|\, x_i - x_j \,|}\Big)^{12} .$$
Here, the $\lambda_{ij}, v_{ij}, w_{ij}, d_{ij}, \delta_{ij}$'s are known constants, depending on the pair of atoms involved (carbon-carbon, carbon-nitrogen, etc.)
- Valence angle: between three atoms i, j, k forming an angle θ_{ijk} (we can write down the value of θ_{ijk} - but it's too heavy), there is an energy
$$A_{ijk}(x_i, x_j, x_k) = \alpha_{ijk}(\theta_{ijk} - \bar\theta_{ijk})^2 \,,$$
where, here again, α_{ijk} and $\bar\theta_{ijk}$ are known constants.

Other types of energies may also be considered: electrostatic, torsion angles, etc. The total energy is then the sum of all these terms, over all pairs/triples/quadruples of atoms. The important thing to understand here, is that this energy can be computed (as well as its derivatives) for any numerical values taken by the variables x_i. And this is true even if these values do not correspond to any reasonable configuration; simply, the resulting energy will then be unreasonably large (if the model is reasonable!); the optimization process, precisely, will aim at eliminating these values.

Note that the objective function is disagreeable: First with its many terms, it is long to compute. Second, with its strong nonlinearities, it does not enjoy some nice properties useful for optimization: it is definitely not quadratic, and not even convex. Actually, in most examples there are many equilibrium points X^* (local minima); this is why the only hope is to *refine* a specific one: by assumption, some estimate X_0 is available, close to the sought "optimal" X^*. Otherwise the optimization algorithm could only find some uncontrolled equilibrium, "by chance". Note also that nowaday's "interesting" molecules have 10^3 atoms and more, so we have to fight against numerical difficulties...

Meteorology [BGLS §1.2.2] Consider the problem of forecasting the weather, i.e. of knowing the state of the atmosphere in the future. For this, let $p(z, t)$ be the state of the atmosphere at point $z \in \mathbb{R}^3$ and time $t \in [0, 7]$ (with t expressed in days, assuming a forecast over one week); p is actually a vector made up of pressure, wind speed, humidity... The evolution of p along time can be modelled: avoiding technicalities, fluid mechanics tells us that
$$\frac{\partial p}{\partial t}(z, t) = \Phi(p(z, t))\,, \tag{2}$$
where Φ is a certain differential operator. For example, (2) could be the Navier-Stokes equation, but approximations are generally introduced.

Once our model Φ is chosen, it "suffices" to integrate (2). For this, initial conditions are needed (the question of boundary conditions is neglected here; for example, we shall say that they are periodicity conditions, (2) being integrated on the whole earth). Here comes optimization, in charge of estimating $p(\cdot, 0)$ via an *identification* process, which we explain roughly.

In fact, the available information also contains all the meteorological observations collected in the past, say during the preceding day. Let us denote by $\omega = \{\omega_i\}_{i \in I}$ these observations. To fix ideas, we could say that each ω_i represents the value of p at a certain point (z_i, t_i). To take this data into account, a natural idea is to consider the problem
$$\min_p \,|\, p - \omega \,|\,, \tag{3}$$
(2) being considered as a constraint (called in this context the *state equation*; in fact we have here an *optimal control* problem, in which the objective function depends on a *state* (here p) evolving along time).

At this point, it is a good idea not to view (2), (3) as a nonlinearly constrained optimization problem, but rather as an unconstrained one. In fact, call $x(z) = p(z, -1)$ the state of the atmosphere at z, at initial time $t = -1$. A fundamental remark is then: assuming x to be known, (2) gives $p = p_x$ unambiguously, and hence the objective value in (3) as well: the unknown p_x depends on the variable x *only*. Our problem can therefore be formulated as $\min_x \mid p_x - \omega \mid$, which means:
– to minimize with respect to x (unconstrained variable)
– the function defined by (3),
– where $p = p_x$ is obtained from (2)
– via the initial condition $p(\cdot, -1) = x$.

NB. Just as p is called the state, x – i.e. $p(\cdot, -1)$ here – is the *control* variable: there is a well-defined mapping

$$\text{control} \quad \mapsto \quad \text{state} \quad \mapsto \quad \text{objective function,}$$

and this is characteristic of optimal control problems.

1.2 General Principles of Resolution [BGLS §1.3]

Our problem (1) will be solved via an algorithm which constructs iteratively $x_1, x_2, \ldots, x_k, \ldots$ To obtain the next iterate, the algorithm needs to know some information concerning f; essentially, the numerical value $f(x)$ for each value of x; often, its gradient $\nabla f(x)$ as well. This information is computed in a *black box* (subprogram), independent of the selected algorithm. This subprogram can be called *simulator*, since it simulates the behaviour of the problem under the action of the decision variables (optimal or not).

Hence a computer program solving an optimization problem is made up of *two distinct parts*:

- one is in charge of managing x and is the algorithm proper; call it (A), as Algorithm; it is generally written by a mathematician, specialized in optimization;
- the other, the simulator, performs the required calculations for each x decided by (A); it is generally written by a practitioner (physicist, economist, etc.), the one who wishes to solve the specific optimization problem. For example, the simulator in §1.1 integrates the state equation (2), computes the deviation from the observations (as well as the gradient, see below), and passes the result to the Algorithm in charge of solving the dual problem.

Another fundamental thing to understand here is the following: for any problem considered, the only information available for f is the result of a numerical calculation, generally complicated (think of the Navier-Stokes equation of Example 1.1). Hence, (A) has to proceed by "trial and error": it assigns trial values to x, which it corrects upon observation of the answer from the simulator; and this will essentially make up one iteration of the optimization process.

Now the current iteration k of an optimization algorithm is made up of two phases: to compute a direction, and to perform a line-search.

- Computing a direction: f is replaced by a model f_k, which is simpler; then f_k is minimized to yield a new approximation, call it $x_k + d$.
- Line-search: a stepsize $t > 0$ is computed so that $f(x_k + td) < f(x_k)$.
- The new iterate is then $x_{k+1} = x_k + td$.

The descent property $f(x_k + td) < f(x_k)$ ensures in particular *stability*, a privilege of optimization methods, as compared for example with equation solvers. Beware that the direction is computed by minimizing (usually accurately) an *approximation* of f. The stepsize t is computed by observing the *true* f on the restriction of $x \in \mathbb{R}^n$ to the half-line $\{x_k + td\}_{t \in \mathbb{R}_+}$ (x_k and d fixed). All this will be seen in detail in §2 below.

1.3 Generalities on Convergence [BGLS §1.5]

For an optimization algorithm generating a sequence (x_k), two types of convergence are relevant.

- Global convergence. In optimization, an algorithm is said to converge *globally* when $\liminf |\nabla f(x_k)| = 0$ for any initial iterate x_1. This property guarantees that the stopping test "$|\nabla f(x_k)| \leqslant \varepsilon$?" will occur for sure, for any $\varepsilon > 0$.
- Local convergence. Now assume x_k has a limit x^*; one wants to know at what speed $\delta_k := |x_k - x^*|$ tends to 0. The interesting property is the so-called *superlinear convergence*, which means $\frac{\delta_{k+1}}{\delta_k} \to 0$.

In particular, we say that *quadratic convergence* holds when $\delta_{k+1} = O(\delta_k^2)$; roughly, this means that the number of exact digits doubles at each iteration (for k large enough).

Throughout, we will use the notation g for the gradient ∇f; and most of the time, g_k will stand for $g(x_k) = \nabla f(x_k)$.

1.4 Computing the direction: general principles

Basically, the direction is computed by approximating $g(x) := \nabla f(x)$ near x_k. The simplest scheme is to minimize the first-order approximation $f(x_k)+\nabla f(x_k)^\top d$ of f near x_k. The corresponding approximate objective function $\nabla f(x_k)^\top d$ has no minimum at finite distance, unless a normalization is appended. Thus, a steepest-descent direction associated with a norm $|\cdot|$ is a solution of

$$\min \nabla f(x_k)^\top d\,, \quad |d| \leqslant 1\,.$$

Taking the Euclidean norm $|\cdot|$ for $|\cdot|$ results in the gradient method $d = -\nabla f(x_k)$ (up to normalization).

2 Line-Searches [BGLS Chap. 3]

As explained in §1.2, one iteration of an optimization algorithm is made up of two steps; we now study the second step: computing the stepsize. So, we are given:
– the starting point x of the line-search (x is the current iterate x_k);
– the direction of search d;
– a merit-function $t \mapsto q(t)$, defined for $t \geqslant 0$, representing $f(x+td)$.

In the following, we will always suppose $\boxed{q'(0) = \nabla f(x)^\top d < 0}$.

2.1 General Scheme

The whole business of a line-search algorithm is to define a test with three possible outputs which, given $t > 0$, answers whether

a) t is satisfactory
b) t is too large
c) t is too small.

This test is performed upon observation of $q(t)$, and possibly of $q'(t)$.

Now we will call t_L a too small t (on the left of a desired t), t_R a too large t (on the right of a desired t). To initialize the search, 0 is obviously a t_L, while t_R can be initialized to $+\infty$, whatever it means. Schematically, the algorithm is then the following:

Schematic line-search algorithm

```
Step 0. Start from an initial $t > 0$. Initialize $t_L = 0$ and $t_R = +\infty$.
Step 1. Test $t$;
    if a) terminate;
    if b) declare $t_R = t$ and go to 2;
    if c) declare $t_L = t$ and go to 2.
Step 2
    If no real $t_R$ has been found ($t_R = +\infty$), compute a new $t > t_L$.
    Else compute a new $t \in ]t_L, t_R[$.
    Loop to 1.
```

Thus, the line-search algorithm is a sequence of interpolations, reducing the *bracket* $[t_L, t_R]$, possibly preceded by a sequence of extrapolations (as long as $t_R = +\infty$).

The observations below are straightforward, but fundamental for a good understanding of the mechanism.

– Extrapolations are performed until a finite t_R is found (which may happen at the first try).

- Then the interpolation phase starts.
- In any case, t_L increases each time it is modified,
- and t_R decreases each time it is modified;
- but there always holds $t_L < t_R$.

2.2 Modern line-search: Wolfe's rule

A line-search must terminate as soon as possible, since it is a subalgorithm within the optimization process. In particular, finding an optimal $t > 0$ would be absurd: what we want is to minimize f, not q. From this point of view, the method admitted as the most efficient is the following, commonly called the *Wolfe* line-search. Two coefficients $0 < m_1 < m_2 < 1$ are chosen, and cases a) b) c) are the following:

a) $q(t) \leqslant q(0) + m_1 t q'(0)$	and	$q'(t) \geqslant m_2 q'(0)$	(then terminate);
b) $q(t) > q(0) + m_1 t q'(0)$			(then $t_R = t$);
c) $q(t) \leqslant q(0) + m_1 t q'(0)$	and	$q'(t) < m_2 q'(0)$	(then $t_L = t$).

Note: in (rare) occasions, computing the gradient in the simulator is much more expensive than the function (in terms of CPU). For such problems, a drawback of Wolfe's line-search is that q' – hence ∇f – is needed at each cycle. An alternative line-search exists, called Goldstein & Price, in which $q'(t)$ (the slope at t) is replaced by $(q(t) - q(0))/t$ (the average slope between 0 and t. It is written

a) $q(t) \leqslant q(0) + m_1 t q'(0)$	and	$q(t) \geqslant q(0) + m_2 q'(0)$	(then terminate);
b) $q(t) > q(0) + m_1 t q'(0)$			(then $t_R = t$);
c) $q(t) \leqslant q(0) + m_1 t q'(0)$	and	$q(t) < q(0) + m_2 q'(0)$	(then $t_L = t$).

2.3 Convergence

First one must make sure that the above (sub)algorithm terminates. For this, the new trial stepsize in the scheme 2.1 – call it t_+ – must satisfy some consistency property: for example, t_+ should not be unduly close to t_R, otherwise the bracket will not be properly reduced. Thus, consistency of t_+ means two things:

- Extrapolations should "significantly" increase t; say $t_+ \geqslant 2t_L$.
- Interpolations should "significantly" decrease $[t_L, t_R]$; say $t_+ \in [t_L + \delta, t_R - \delta]$ with $\delta = 0.1(t_R - t_L)$.

Theorem [Consistency of Wolfe line-search, BGLS Thm. 3.7] Suppose $q \in C^1$, $q'(0) < 0$, and t_+ is consistently chosen. Then Wolfe's line-search
– either produces $t_L \to +\infty$ with $q(t_L) \to -\infty$,
– or produces a) after finitely many cycles.

The next question is whether the resulting sequence (x_k) converges to a minimum point. Of course, convergence cannot hold independently of the choice of the direction (the line-search will be helpless if d_k is "too orthogonal" to the gradient). The angle between the direction and the gradient thus appears as essential, and we set

$$c_k := \frac{-g_k^\top d_k}{|\, g_k \,| \,.\, |\, d_k \,|} \,.$$

Theorem [Convergence Wolfe line-search, BGLS Thm. 3.9] With the above notation, let an algorithm generate directions satisfying $\sum_{k=1}^{+\infty} c_k^2 = +\infty$ and use Wolfe's line-search. Assume that g is Lipschitz continuous on the slice $\{x : f(x) \leqslant f(x_1)\}$. Then: either the objective-function tends to $-\infty$, or $\liminf \,|\, g_k \,| = 0$.

Note that this theorem guarantees in particular the good behaviour of an optimization method that combines gradient descent and Wolfe line-search. This explains the numerical experiments of TP2. Nevertheless, we can't be completely content with just a convergent algorithm. We have to demand some efficiency. Here "efficiency" means minimizing the number of calls to the simulator (since in pratice we have no control in what happens inside the simulator).

In pratice A last remark: an optimization program fails rather often (for example first try, before serious debug). Experience indicates that, in 90% of cases, the mistake is not in the Algorithm proper but in the simulator, more precisely in the gradient computation. Such a mistake can be detected rather reliably, upon observation of Wolfe's line-search. If the gradient is bugged, the line-seach eventually produces the following paradoxical behaviour: a sequence $\{t = t_R\}$ is produced, tending to 0 with $q(t)$ staying stubbornly larger than $q(0)$, while $q'(t)$ stays stubbornly negative.

3 Newtonian Methods [BGLS Chap. 4]

Now comes the *most important* approach (by far) to compute a descent direction at each iteration of a minimization algorithm: the quasi-Newton method, defined in §3.3 below. To use another direction cannot be considered without a serious motivation; this has been true for decades and will probably remain so for several more years.

3.1 Preliminaries

To solve the optimality condition $g(x) = 0$, the Newton principle is as follows. Starting from the current iterate x_k, replace g by its linear approximation:

$$g(x_k + d) = g(x_k) + g'(x_k)d + o(\,|\,d\,|\,)$$

where $g'(x_k)$ is the Jacobian of g at x_k. We then neglect the term $o(|d|)$; this gives the linearized problem $g(x_k) + g'(x_k)d = 0$. Its solution is $d^N = -[g'(x_k)]^{-1}g(x_k)$, producing the Newton iterate $x^N = x_k + d^N$.

In the case of an optimization problem, g is the gradient of f, $g' = \nabla^2 f$ is its Hessian. Just as g was approximated to first order, f can be approximated to second order:

$$f(x_k + d) = f(x_k) + g(x_k)^\top d + \frac{1}{2}d^\top \nabla^2 f(x_k)d + o(\,|\,d\,|\,^2)\,.$$

The quadratic approximation thus obtained is minimized (if $\nabla^2 f(x_k)$ is positive definite) when its gradient vanishes: $g(x_k) + \nabla^2 f(x_k)d = 0$. We realize an evidence: Newton's method on $\min f(x)$ is Newton's method on $g(x) = 0$.

The big advantage of this method is well known: it converges very fast.

Theorem [Local convergence of Newton] Let f be C^2 near a solution x^*, where $\nabla^2 f(x^*)$ is positive definite. Then the convergence of Newton's method is superlinear. If, in addition, $f \in C^3$, this convergence is quadratic.

This theorem implies by no means global convergence. It simply says that, if x is close to the solution, then x^N is infinitely closer. In fact, drawbacks of Newton's method are also well-known:

- in general, it diverges violently;
- in addition, it requires to compute the Hessian, and then solve a linear system; this is heavy;
- in our situation where g is the gradient of a function to be *minimized*, another drawback is that the sequence (x_k) will probably rush to the closest stationary point, possibly a local maximum.

3.2 The Basic Idea

The following ideas will be used to suppress the drawbacks above.

- In optimization, stability can be enforced by the requirement $f(x_{k+1}) < f(x_k)$; and this is the duty of the line-search. Then it suffices to consider the Newton increment d^N as a *direction*, along which a line-search will be performed to decrease the function $q(t) = f(x_k + td^N)$. This will take care of the first and third drawbacks.

  However, this line-search will be possible only if $q'(0) = g_k^\top d^N < 0$ (see §2). Using the definition of d^N, this means that $g_k^\top \nabla^{-2} f_k^{-1} g_k$ must be positive, i.e. in practice $\nabla^2 f(x_k)$ positive definite; if it is not, something more must be done.

- Consider now the second drawback of Newton's method. Rather than computing explicitly $\nabla^2 f$ at each iteration, and solving the corresponding linear system, we can approximate directly $\nabla^{-2} f$ by a matrix W, which we deliberately choose symmetric positive definite. Then, alongside with the descent process on f, an identification process of the Hessian (or rather its inverse) is performed.

This is the *quasi-Newton* idea; it results in a general algorithm of the following form. Hereafter we drop the index k; and a superscript "+" will mean $k+1$.

Schematic quasi-Newton algorithm

Step 0. An initial iterate x and stopping tolerance ε are given; an initial matrix W, positive definite, is also chosen. Compute the initial gradient $g = g(x)$.

Step 1. If $|\,g\,| \leqslant \varepsilon$ stop.

Step 2. Compute $d = -Wg$.

Step 3. Make a line-search initialized on $t = 1$, to obtain the next iterate $x_+ = x + td$ and its gradient $g_+ = g(x_+)$.

Step 4. Compute the new matrix W_+ for the next iteration and loop to 1.

The ingredients characterizing this method are therefore: the line-search (which will be of course that of Wolfe, §2.2), the initial matrix W (which can be the identity), the computation of W_+ in Step 4. Let us explain how this last matrix is computed. In the sequel, we will use the notation

$$s = s_k = x_{k+1} - x_k \quad \text{and} \quad y = y_k = g_{k+1} - g_k$$

(observe that s and y are known when W_+ must be computed).

Knowing that we want to approximate a symmetric matrix (an inverse Hessian) and to obtain descent directions, W_+ is of course required to be *symmetric positive definite.*

Besides, to give W_+ a chance to approximate what we want, W_+ is required to satisfy $W_+ y = s$, called the *quasi-Newton*, or *secant* equation. Its explanation is as follows: the mean-value G of $\nabla^2 f$ between x and x_+ satisfies $y = Gs$. The quasi-Newton equation has therefore the effect of forcing $W_+ \simeq G^{-1}$, in the sense that these two matrices have the same action on y, a subspace of dimension 1.

Of course, the two above requirements leave infinitely many possible *quasi-Newton matrices.*

3.3 Quasi-Newton Methods

A quasi-Newton method is the realization of the schematic algorithm 3.2, where the matrices W_k are computed sequentially: $W_+ = W + B$, the corrections B being chosen so that

(i) W_k is symmetric positive definite for all k,

(ii) the quasi-Newton equation $W_+ y = s$ is satisfied for all k.

Among all the possible corrections, stability reasons lead us to the additional requirement:

(iii) each B is minimal in some sense.

This still leaves a large range of possibilities, depending on the sense chosen in (iii). At present, a consensus is obtained for a method found independently, and using different arguments, by C. Broyden, R. Fletcher, D. Goldfarb, D. Shanno:

$$W_+ = W_+^{BFGS} = W - \frac{sy^\top W + Wys^\top}{y^\top s} + \left[1 + \frac{y^\top W y}{y^\top s}\right] \frac{ss^\top}{y^\top s}\,.$$

Check that it is symmetric and satisfies the quasi-Newton equation. Also, observe from the quasi-Newton equation $s = W_+ y$ that positive definiteness requires $y^\top s > 0$. This last condition is actually sufficient:

Theorem Suppose W is positive definite. Then $y^\top s > 0$ is a necessary and sufficient condition for W_+^{BFGS} to be positive definite.

An interesting point in this result is that the property $y^\top s > 0$ means $g_+^\top s > g^\top s$, and this is just the same as the second part of Wolfe's rule of §2.2: $q'(t) \geqslant m_2 q'(0)$.

Remark If $n = 1$, the quasi-Newton equation defines a unique W_+ by $W_+ = s/y$. Starting from two initial iterates x_1 and x_2, the algorithm becomes

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{g_k - g_{k-1}} g_k$$

known as the secant method, or "regula falsi": the tangent to the graph of g (which is used in Newton's method) is replaced by the secant between x and x_-.

The opposite case (big n) also deserves comment. A big drawback of Newtonian methods (including the present quasi-Newton variant) is the necessity of storing an $n \times n$ matrix (what if $n = 10^6$, as in the meteo problem of §1.1?) Yet, as long as k is small, computing the direction d_k of a quasi-Newton method needs only $2k$ vectors s_i and $y_i, i = 1, \dots, k-1$. In the case of a really large-scale problem, at least a few iterations can be performed, just by developing the product $W_k g_k$ (as a function of these k vector pairs), instead of computing explicitly the whole matrix W_k. It is indeed possible to take advantage of this remark, which gives birth to *limited memory* quasi-Newton methods.

3.4 Convergence

Rather unexpectedly, it seems impossible to prove global convergence without convexity, even though such a property seems irrelevant for the matter.

Theorem [Global convergence of BFGS, BGLS Thm. 4.9] Suppose f is convex with a Lipschitzian gradient on the domain $\{x : f(x) \leqslant f(x_1)\}$. Then the BFGS algorithm with Wolfe's line-search and W_1 positive definite satisfies:
– either the objective function tends to $-\infty$,
– or $\liminf |\, g(x_k) \,| = 0$.

Remember that Newtonian methods are designed to converge fast; it is therefore important to study their speed of convergence. Recall that speed of convergence usually makes additional assumptions.

Theorem [Superlinear convergence of BFGS + Wolfe] Let BFGS+Wolfe generate a sequence (x_k) converging to x^* where $\nabla^2 f$ is positive definite and Lipschitz continuous. Assume $0 < m_1 < \frac{1}{2}$, $m_2 > 0$, and the stepsize $t = 1$ is tried first, at each iteration. Then the convergence is superlinear: $|\, x_+ - x^* \,| = o(\,|\, x - x^* \,|\,)$.

4 Lagrangian Duality and Convex Analyis

4.1 The Problem and the General Approach

Consider a constrained optimization problem

$$\left|\begin{array}{ll} \sup \varphi(u) & u \in U\,, \\ c_j(u) = 0 & \text{for } j = 1, \dots, m\,, \end{array}\right. \tag{4}$$

which we will call the *primal problem.*

So far, we are not assuming any structure in U whatsoever. This implies in particular that the objective- and constraint-functions have no structure either, such as convexity, or a fortiori differentiability: in U, these words are meaningless for the moment. It is useful to see how far the theory can be developed in abstracto.

We define the *Lagrangian*

$$L(u, \lambda) := \varphi(u) - \sum_{j=1}^{m} \lambda_j c_j(u) = \varphi(u) - \lambda^\top c(u)\,,$$

depending on the *dual* variable $\lambda \in \mathbb{R}^m$, in addition to the primal variable $u \in U$.

Assumption (practical) The *Lagrange optimization problem*:

$$\sup_{u \in U} L(u, \lambda)\,, \tag{5}$$

where λ is fixed in $\mathbb{R}^m$, is considerably simpler than the primal problem.

What we want is then to find some suitable value of λ such that the associated Lagrange problem provides a solution of the primal problem.

The *dual problem* of finding an appropriate λ is the whole business of *duality theory*, which itself heavily relies on *convex analysis*.

4.2 An important example: Electrical production

Consider a set I of production plants. Call $u_i = (u_i^1, \dots, u_i^T)$ the production vector of plant i over the period $1, \dots, T$ and let $c_i(u_i)$ be the resulting production cost. At each time period t, the demand d^t is known; assume for simplicity that we want to satisfy this demand exactly. Thus, we have to solve

$$\left| \begin{array}{l} \min \sum_{i \in I} c_i(u_i)\,, \quad u_i \in U_i\,, \\ \sum_{i \in I} u_i^t = d^t \text{ for } t = 1, \dots, T\,. \end{array} \right.$$

The sets U_i represent feasible production plannings; they can be fairly complicated and diverse, depending on each plant type (nuclear, hydraulic,. . .)

We are faced with a large-scale, nonlinear, mixed integer programming problem. Dualizing the demand constraints, we obtain the Lagrangian

$$L(u, \lambda) := \sum_{i \in I} c_i(u_i) + \sum_{t=1}^{T} \lambda^t \big(\textstyle\sum_{i \in I} u_i^t - d^t\big) = \displaystyle\sum_{i \in I} \big[c_i(u_i) + \lambda^\top u_i\big] - \lambda^\top d\,,$$

to be minimized over $u \in \prod U_i$. Clearly enough, the Lagrangian problem can be decomposed: to solve it, it suffices to optimize each power plant separately, i.e. to solve $\min_{u_i \in U_i} c_i(u_i) + \lambda^\top u_i$ for each $i \in I$.

4.3 Elementary Results

We return to problem (4) of Subsection 4.1. The optimal value of the Lagrangian problem (5) is a number depending on λ. It is a fundamental object in duality theory, called the *dual function*:

$$\theta(\lambda) := \sup_{u \in U} L(u, \lambda)\,.$$

Theorem [Weak Duality] For all $\lambda \in \mathbb{R}^m$ and all u primal-feasible, there holds $\theta(\lambda) \geqslant \varphi(u)$.

Thus, each value of the dual function gives an upper bound on the primal optimal value; a sensible idea is then to find the best such upper bound, i.e. the minimal value of θ. Another observation is as follows: we want to obtain a u_λ satisfying $c(u_\lambda) = 0$; then $\theta(\lambda) = L(u_\lambda, \lambda) = \varphi(u_\lambda)$; in view of weak duality, this implies that $\theta(\lambda)$ reaches its minimal value. This makes two arguments showing that the *dual problem* is

$$\inf \{\theta(\lambda) \;:\; \lambda \in \mathbb{R}^m\}\,. \tag{6}$$

The above result has important consequences, based on the simple observation that $L(u, \lambda) = \varphi(u)$ if u is feasible.

Corollary If, for some $\lambda \in \mathbb{R}^m$ the Lagrangian problem happens to have a solution u_λ which is feasible ($c(u_\lambda) = 0$) then this u_λ is a primal optimal solution.

To solve the primal problem, we of course *must* solve the system of equations $c_\lambda = c(u_\lambda) = 0$, where u_λ maximizes the Lagrangian. The above result then says that it *suffices* to solve this system. As stated, the problem is not simple: the mapping $\lambda \mapsto c_\lambda$ is not even well-defined since u_λ is ambiguous (the Lagrangian need not have a unique maximum). But a second consequence of weak duality is that finding λ is indeed an optimization problem:

Corollary If $\bar{\lambda}$ is such that the associated Lagrange problem is maximized at a primal-feasible $\bar{u}$ (hence $\bar{u}$ is a primal optimal solution) then $\bar{\lambda}$ is also a solution of the dual problem.

4.4 Basic Convex Analysis [HULL Chaps III,IV,VI]

We will see that the dual problem (6) is indeed convex, which makes it necessary to introduce some rudiments of convex analysis.

Definition The set $C \subset \mathbb{R}^n$ is said to be convex if $\alpha x + (1-\alpha)x'$ is in C whenever x and x' are in C, and $\alpha \in]0,1[$ (or equivalently $\alpha \in [0,1]$).

Proposition Let $\{C_j\}_{j \in J}$ be an arbitrary family of convex sets. Then $C := \cap\{C_j : j \in J\}$ is convex.

Note that, if the C_j's are closed, then C is also closed.

Definition A function $\theta : \mathbb{R}^m \to \bar{\mathbb{R}}$, not identically $+\infty$, is said to be convex when, for all $(\lambda, \lambda') \in \mathbb{R}^m \times \mathbb{R}^m$ and all $\alpha \in]0,1[$, there holds $\theta(\alpha\lambda + (1-\alpha)\lambda') \leqslant \alpha\theta(\lambda) + (1-\alpha)\theta(\lambda')$, considered as an inequality in $\bar{\mathbb{R}}$.

Definitions The domain of a function θ from $\mathbb{R}^m$ to $\bar{\mathbb{R}}$ ($\theta \not\equiv +\infty$) is the nonempty set $\operatorname{dom}\theta := \{\lambda \in \mathbb{R}^m : \theta(\lambda) < +\infty\}$.

The epigraph of θ is the nonempty set $\operatorname{epi}\theta := \{(\lambda, r) \in \mathbb{R}^m \times \mathbb{R} : r \geqslant \theta(\lambda)\}$.

Proposition Let $\theta : \mathbb{R}^m \to \bar{\mathbb{R}}$ be not identically $+\infty$. The two properties below are equivalent:

- θ is convex;
- its epigraph is a convex set in $\mathbb{R}^m \times \mathbb{R}$.

The set of convex functions from $\mathbb{R}^m$ to $\bar{\mathbb{R}}$ will be denoted by $\operatorname{conv}\mathbb{R}^m$.

Proposition For $\theta : \mathbb{R}^m \to \bar{\mathbb{R}}$ (not necessarily convex), the following three properties are equivalent:

(i) θ is lower semi-continuous on $\mathbb{R}^m$ (i.e. $\liminf_{\mu \to \lambda} \theta(\mu) \geqslant \theta(\lambda)$, for all $\lambda \in \mathbb{R}^m$);
(ii) $\operatorname{epi}\theta$ is a closed set in $\mathbb{R}^m \times \mathbb{R}$;
(iii) the sublevel-sets $\mathrm{S}_r(\theta) := \{\lambda : \theta(\lambda) \leqslant r\}$ are closed (possibly empty) for all $r \in \mathbb{R}$.

Definition The function $\theta : \mathbb{R}^m \to \bar{\mathbb{R}}$ is said to be closed if it is lower semi-continuous everywhere, or if its epigraph is closed, or if its sublevel-sets are closed. The set of closed convex functions from $\mathbb{R}^m$ to $\bar{\mathbb{R}}$ will be denoted by $\overline{\operatorname{conv}}\mathbb{R}^m$.

Proposition Let $\{\theta\}_{j \in J}$ be an arbitrary family of convex [resp. closed convex] functions. If there exists λ_0 such that $\sup_j \theta_j(\lambda_0) < +\infty$, then their pointwise supremum $\theta := \sup_{j \in J} \theta_j$ is in $\operatorname{conv}\mathbb{R}^m$ [resp. in $\overline{\operatorname{conv}}\mathbb{R}^m$].

Definition The subdifferential $\partial\theta(\lambda)$ of θ at λ is the set of vectors $g \in \mathbb{R}^m$ satisfying

$$\theta(\mu) \geqslant \theta(\lambda) + g^\top(\mu - \lambda) \quad \text{for all } \mu \in \mathbb{R}^m .$$

The subdifferential is a closed convex set in $\mathbb{R}^m$. Clearly enough, λ minimizes θ if and only if $0 \in \partial\theta(\lambda)$. The subdifferential of a sup-function is an important tool for optimization.

Theorem Let $\theta = \sup_{j \in J} \theta_j$, each θ_j being closed convex. Let $g_j \in \partial\theta_j(\lambda)$, with j such that $\theta(\lambda) = \theta_j(\lambda)$. Then $g_j \in \partial\theta(\lambda)$.

Because the subdifferential is a convex set, any convex combination of active subgradients is still a subgradient. A key issue is whether the converse is true; this is the so-called filling property (see below).

4.5 Properties of the Dual Problem

Beware that dual optima – i.e. solutions of (6) – need not produce primal optima: it may well happen that θ does have a maximum point, while no Lagrangian problem, for any λ, produces any feasible u_λ.

Definition [Duality Gap] The difference between the optimal primal and dual values

$$\inf_{\lambda \in \mathbb{R}^m} \theta(\lambda) - \sup_{u \in U} \{\varphi(u) : c(u) = 0\}$$

is a nonnegative number (weak duality theorem), called the duality gap.

Let λ^* be a dual optimal solution. To say that it produces a primal optimal solution u^* is to say that $L(u^*, \lambda^*) = \theta(\lambda^*)$ and that $c(u^*) = 0$. Then we have $\theta(\lambda^*) = L(u^*, \lambda^*) = \varphi(u^*)$: the duality gap is zero. Conclusion: to say that Lagrangian duality works is to say that there is no duality gap.

Proposition [very important] If not identically $+\infty$, the dual function θ lies in $\overline{\text{conv}}\mathbb{R}^m$. Furthermore, for any u_λ maximizing the Lagrangian at a given λ, the corresponding vector $g := -c(u_\lambda) \in \mathbb{R}^m$ is a subgradient of θ at λ.

A dual solution λ is characterized by $0 \in \partial\theta(\lambda)$; now the subdifferentials of θ lie in the (dual of the) λ-space, i.e. in the space of constraint-values. Indeed, consider the (possibly empty) optimal set in the Lagrange problem: $U_\lambda := \{u \in U : L(u, \lambda) = \theta(\lambda)\}$; in view of the above Proposition, $\partial\theta(\lambda)$ contains the closed convex hull of $\{-c(u) : u \in U_\lambda\}$. The converse inclusion is actually crucial, and motivates the following definition.

Definition [Filling Property] We say that the filling property holds at $\lambda \in \mathbb{R}^m$ when $\partial\theta(\lambda)$ is the convex hull of the vectors $-c(u)$ for u describing U_λ.

This provides an easy description of $\partial\theta$ in terms of primal points: when the filling property holds, to say that $g \in \partial\theta(\lambda)$ is to say that there are
- primal points u_k's maximizing the Lagrangian,
- and corresponding convex multipliers α_k's
- such that $g = -\sum \alpha_k c(u_k)$.

Lemma [HULL, Thm VI.4.4.2] Suppose that U is a compact set, on which φ is upper semicontinuous, and each c_j is continuous. Then the filling property holds at each $\lambda \in \mathbb{R}^m$.

This set of conditions is rather "normal", in that it goes in harmony with two other important properties: existence of an optimal solution in both the primal and Lagrangian problems.

Putting everything together Here comes the conclusion of the whole theory. Indeed suppose the filling property holds. Then a dual optimal solution λ^* is characterized by $0 \in \partial\theta(\lambda^*) = -\overline{\text{conv}} c(U_{\lambda^*})$, which means the existence of
- primal points $u_1, \dots, u_K$ such that $L(u_k, \lambda^*) = \theta(\lambda^*)$, $k = 1, \dots, K$,
- and convex multipliers $\alpha_1, \dots, \alpha_K$ ($\alpha_k \geqslant 0$ and $\sum_k \alpha_k = 1$)
- such that $\sum_k \alpha_k c(u_k) = 0$.

Then make the following assumptions, one after the other:
- If U is a convex set, we can make up the point $u^* := \sum_k \alpha_k u_k$: then $u^* \in U$.
- If each c_j is an affine function, then $c(u^*) = \sum_k \alpha_k c(u_k) = 0$: u^* is primal-feasible.
- If φ is a concave function then $\varphi(u^*) \geqslant \sum_k \alpha_k \varphi(u_k) = \theta(\lambda^*)$: u^* is indeed primal optimal.

All this is illustrated by the two examples of §4.2:
- Lagrangian duality works for the matrix calibration problem, which is convex.
- It has no reason to work for the electrical problem because the U_i's are not convex. Lagrangian duality becomes a heuristic method: it gives a clever bound but more work has to be done to find a solution to the primal problem.

4.6 Exercices de dualisation

Exo 1. Dualiser le problème

$$\left|\begin{array}{l} \min f(x)\,, \quad x \in X \\ h(x) \leqslant 0 \in \mathbb{R}^m \end{array}\right.$$

en utilisant $f(x) + \lambda^\top h(x)$ pour former le lagrangien. On se ramene au cas précédent en posant

$$u := (x, y) \in U := X \times \mathbb{R}^m_+\,, \quad \varphi(x, y) := -f(x)\,, \quad c(x, y) := h(x) + y = 0\,;$$

alors (4) est une formulation équivalente de notre présent problème. Le lagrangien devient $L(x, y, \lambda) := -f(x) - \lambda^\top(h(x) + y)$. La fonction duale garde la même valeur $\theta(\lambda)$ pour $\lambda \geqslant 0$ mais devient $+\infty$ si une coordonnée de λ est négative.

Exo 2. Même technique pour le problème

$$\min f(x) + \frac{\pi}{2}\,|\,h(x)\,|^2\,, \quad x \in X\,.$$

On pose maintenant

$$u := (x, y) \in X \times \mathbb{R}^m\,, \quad \varphi(x, y) := -f(x) - \frac{\pi}{2}\,|\,y\,|^2\,, \quad c(x, y) := h(x) - y = 0\,.$$

Maximiser le lagrangien $L(x, y, \lambda) := -f(x) - \frac{\pi}{2}\,|\,y\,|^2 - \lambda(h(x) - y)$ ne fait qu'ajouter $\frac{1}{2\pi}\,|\,\lambda\,|^2$ à la fonction duale. Même question pour $\max f(x)$, $g(x) \leqslant 0 \in \mathbb{R}^m$, $x \in X$.

Exo 3. Écrivant un programme linéaire sous forme standard: $\min c^\top x$, $Ax = b$, $x \geqslant 0$, faire 2 schémas de dualité, l'un en dualisant toutes les contraintes, l'autre en ne dualisant que $Ax - b = 0$. Obtenir le 2nd à partir du 1er.

5 Algorithms for (nondifferentiable) convex optimization

The problem considered here is: how to minimize a convex function $\theta(\lambda)$ for $\lambda \in \mathbb{R}^m$, given a simulator which, for given $\lambda \in \mathbb{R}^m$, computes the value $\theta(\lambda)$ and a subgradient $g \in \partial\theta(\lambda)$. We will often denote by g_λ this subgradient. The situation is therefore comparable to "ordinary" optimization, except that the mapping $\lambda \mapsto g_\lambda$ is not continuous, and g_λ need not be 0 even if λ minimizes θ.

A minimization algorithm will construct a (hopefully minimizing) sequence (λ_k) and we will use the notation g_k for g_{λ_k}.

5.1 Subgradient Optimization

The motivation for this method is that, for any $g \in \partial\theta(\lambda)$, $\lambda - tg$ is closer to any minimum point if $t > 0$ is small enough. Controlling t is not possible by an "online" algorithm similar to a line-search; in fact $-g(= -g_\lambda)$ need not be a descent direction for θ at λ. On the other hand, "off-line" controls are possible. A precise result is in fact as follows:

Theorem [HULL Thm XII.4.1.4] For θ convex and finite everywhere, consider the algorithm $\lambda_{k+1} = \lambda_k - t_k g_k$, where the stepsize t_k satisfies the following properties:

$$t_k = \frac{\tau_k}{|\,g_k\,|}\,, \quad \tau_k \to 0\,, \quad \sum_{k=1}^{+\infty} \tau_k = +\infty\,.$$

Then $\liminf \theta(\lambda_k) = \inf \theta$.

Naturally $|\,\lambda_{k+1} - \lambda_k\,| = \tau_k$ forms a divergent series, and this implies that the sequence (λ_k) cannot converge with linear rate. On the other hand the subgradient method has given birth to the celebrated *ellipsoid* method, which has a high theoretical interest in combinatorial optimization and complexity theory.

5.2 The method of Kelley, or Cheney-Goldstein, or cutting planes

A totally opposite idea uses the *cutting-plane* model of θ, namely the piecewise linear function

$$\hat{\theta}_K(\lambda) := \max\left\{\theta(\lambda_k) + g_k^\top(\lambda - \lambda_k) \;:\; k = 1, \dots, K\right\}.$$

This (convex) function is lower than the true θ. If we bet that it is an accurate model of θ, then it makes sense to take λ_{K+1} minimizing it, i.e. an optimal solution of

$$\min\{\hat{\theta}_K(\lambda) \;:\; \lambda \in \mathbb{R}^m\}.$$

Clearly enough, we have $\hat{\theta}_K(\lambda_{K+1}) \leqslant \hat{\theta}_K(\lambda) \leqslant \theta(\lambda)$ for all $\lambda \in \mathbb{R}^m$. The *gap* $\theta(\lambda_{K+1}) - \hat{\theta}_K(\lambda_{K+1})$ is therefore a measure of optimality for λ_{K+1}: indeed

$$\theta(\lambda_{K+1}) - \hat{\theta}_K(\lambda_{K+1}) \leqslant \varepsilon \quad \Longrightarrow \quad \theta(\lambda_{K+1}) \leqslant \inf\theta + \varepsilon\,.$$

A convergence result can be mentioned:

"Theorem" [HULL Thm XII.4.2.3] With a suitable compactness assumption, 0 is a cluster value of the sequence of gaps: $\liminf \theta(\lambda_{K+1}) - \hat{\theta}_K(\lambda_{K+1}) = 0$.

This method is due to Kelley and Cheney-Goldstein. It suffers serious drawbacks and may converge very slowly. Note also that a *starter* is needed: $\hat{\theta}_K$ need not be bounded from below; such is the case for example when $K = 1$.

5.3 Bundle methods

To accelerate the above process, bundle methods take a *stability center* $\hat{\lambda}$ and penalize the deviation from it. In other words, λ_{K+1} is the optimal solution of

$$\min_{\lambda \in \mathbb{R}^m} \left\{\hat{\theta}_K(\lambda) + \frac{1}{2t}\,|\,\lambda - \hat{\lambda}\,|^2\right\}. \tag{7}$$

Here the "spring strength" $t > 0$ may be set to 1; but numerical efficiency requires to tune it at each iteration.

The above optimization problem producing λ_{K+1} is indeed linear-quadratic (quadratic cost, affine constraints), as can be seen by introducing a vertical variable r: then it becomes

$$\begin{cases} \min r + \frac{1}{2t}\,|\,\lambda - \hat{\lambda}\,|^2 & (\lambda, r) \in \mathbb{R}^m \times \mathbb{R} \\ r \geqslant \theta(\lambda_k) + g_k^\top(\lambda - \lambda_k) & k = 1, \dots, K\,. \end{cases}$$

As for the stability center, the simplest is to take as $\hat{\lambda}$ the best among the K iterates:

$$\hat{\lambda} = \lambda_{\hat{k}}\,, \quad \text{where} \quad \theta(\lambda_{\hat{k}}) \leqslant \theta(\lambda_k)\,,\; k = 1, \dots, K\,.$$

But to obtain a convergent algorithm, one proceeds analogously to line-searches (Armijo test), requesting a strict decrease $\theta(\hat{\lambda}) - \theta(\lambda_{K+1})$.

This method does not suffer from the drawbacks of the cutting planes. Moreover it relies on firm theoretical grounds (skipped in these notes - see [HULL]) and it also proves to be quite efficient in pratice. Note eventually that it appeals for further improvements.