Unconstrained and constrained optimization algorithms
Lecture notes

TAF MCE - UE Numerical Methods

thierry.chonavel@imt-atlantique.fr

Contents

In these notes we supply additional material about unconstrained and constrained optimization algorithmsTo keep them concise, we focus more on their derivation than on convergence issues for which discussions can be found in references. We assume, when necessary, that f is a real valued convex function and that it belongs to $C^2(\mathbb{R}^n)$.

1 Unconstrained optimization

1.1 First order methods

Local descent To find the local minimum of a function $f : \mathbb{R}^n \to \mathbb{R}$ we design a sequence $(\mathbf{x}_k)_{k\geq 0}$ with $\mathbf{x}_{k+1} = \mathbf{x}_k + \rho_k \mathbf{d}_k$ where $\mathbf{d}_k$ is the local search direction and ρ_k $(k > 0)$ the stepsize.
Assuming that $\mathbf{d}_k$ is a descent direction at point $\mathbf{x}_k$ (in particular we must have $\mathbf{d}_k^T \nabla f(\mathbf{x}_k) < 0$), **line search** consists in looking for the optimal stepsize $\rho_k = \arg\min_\rho f(\mathbf{x}_k+\rho\mathbf{d}_k)$, while **approximate line search** is often based on the following approach: noting that

$$f(\mathbf{x}_k + \rho\mathbf{d}_k) = f(\mathbf{x}_k) + \rho\mathbf{d}_k^T \nabla f(\mathbf{x}_k) + o(\rho), \tag{1}$$

to guarantee the decrease of f for small ρ we require that

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \alpha\rho\mathbf{d}_k^T \nabla f(\mathbf{x}_k), \tag{2}$$

where α is fixed with $0 < \alpha < 1$. This condition, known as **Armijo condition**, becomes valid as ρ goes down to 0, but we would like the move from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ not to be too small. Thus, we progressively decrease ρ from an initial value until Armijo condition is satisfied. This leads to the **backtracking algorithm** summarized in algorithm 1.
Further refinements of this kind of approach can be obtained by also considering the evolution of $\nabla f(\mathbf{x}_k)$ (**Wolfe conditions** [7])

Stopping conditions for local descent algorithms can be based on the maximum number of iterations, the decrease or relative decrease of f, the evolution of $\mathbf{x}$, the absolute value of the gradient, or combination of these informations.

Algorithm 1 Backtracking algorithm

1: **Set** $0 < \alpha < 1/2$ and $0 < \beta < 1$
2: $\rho = 1$
3: **while** $f(\mathbf{x}_k + \rho\mathbf{d}_k) > f(\mathbf{x}_k) + \alpha\rho\mathbf{d}_k^T \nabla f(\mathbf{x}_k)$ **do**
4: $\quad \rho = \beta\rho$
5: **end while**
6: **Return** $x_{k+1} = \mathbf{x}_k + \rho\mathbf{d}_k$

Gradient algorithms In the case of gradient descent methods, we consider $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ or $\mathbf{d}_k = \frac{-1}{\|\nabla f(\mathbf{x}_k)\|+\varepsilon}\nabla f(\mathbf{x}_k)$ with $0 < \varepsilon \ll 1$ can be introduced to avoid numerical issues. $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ is indeed a descent direction since then, for small ρ,

$$f(\mathbf{x}_k + \rho\mathbf{d}_k) = f(\mathbf{x}_k) - \rho \parallel \nabla f(\mathbf{x}_k) \parallel^2 + o(\rho) < f(\mathbf{x}_k); \tag{3}$$

The **gradient algorithm with optimal stepsize**, is given in algorithm 2. Alternatively one can approximate line search for the stepsize computation by using the backtracking procedure in algorithm 1 or other stepize strategies (a review of stepsize optimization techniques can be found in [4]) and step 4 in 1 would write $\rho \approx \arg\min_r f(\mathbf{x} + r\mathbf{d})$. In the following, this notation will be used to note that either line search or approximate line search is considered.

Algorithm 2 Gradient algorithm (optimal stepsize)

1: **init** $\mathbf{x}_0$
2: **while** stopping condition $\neq$ true **do**
3: $\quad \mathbf{d} = -\nabla f(\mathbf{x})$
4: $\quad \rho = \arg\min_r f(\mathbf{x} + r\mathbf{d})$
5: $\quad \mathbf{x} = \mathbf{x} + \rho\mathbf{d}$
6: **end while**
7: **return** $\mathbf{x}$

For the gradient algorithm with optimal stepsize, we can check that successive descent directions are orthogonal. Indeed,

$$\begin{aligned} \frac{d}{d\rho} f(\mathbf{x}_k + \rho\mathbf{d}_k)\Big|_{\rho=\rho_k} &= \mathbf{d}_k^T \nabla f(\mathbf{x}_k + \rho_k\mathbf{d}_k) \\ &= -\mathbf{d}_k^T \mathbf{d}_{k+1} = 0. \end{aligned} \tag{4}$$

where the last equality stands from the optimality of the stepsize.

Conjugate gradient algorithm The orthogonality of successive gradients in optimal stepsize gradient algorithm that we have just mentioned here above shows that successive approximations of the solution form a zig-zagging trajectory. The conjugate gradient algorithm (CG), inspired from quadratic optimization, helps combatting this effect.
Consider first the case where $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} + c$, where $\mathbf{A} \succ 0$ ($\mathbf{A}$ is positive definite). Assume now a set of nonzero directions $\mathbf{d}_k$, $k \in 0:n-1$, such that $\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j = 0$ for $i \neq j$. Such directions are called **conjugate vectors** with respect to (w.r.t.) $\mathbf{A}$. Conjugate vectors satisfy the following properties:

Theorem 1 *For $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$, where $\mathbf{A} \succ 0$, and a sequence $(\mathbf{d}_k)_{k=0:n-1}$ of conjugate vectors associated to $\mathbf{A}$, we have the following properties:*

1. *$(\mathbf{d}_k)_{k=0:n-1}$ is a basis of $\mathbb{R}^n$.*
2. *$\mathbf{x}^* = \sum_{k=0:n-1} \frac{\mathbf{d}_k^T\mathbf{b}}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}\mathbf{d}_k$ yields the minimum of f.*
3. *For $\mathbf{x}_0 \in \mathbb{R}^n$, the sequence*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{d}_k^T(\mathbf{A}\mathbf{x}_k - \mathbf{b})}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}\mathbf{d}_k,\ k = 0:n-1, \tag{5}$$

yields $\mathbf{x}_n = \mathbf{x}^*$.

proof

1. Let $c_{0:n-1}$ real numbers such that $\sum_{k=0:n-1} c_k\mathbf{d}_k = 0$. Then, for any $i \in 0:n-1$,

$$\mathbf{d}_i^T\mathbf{A}(\sum_{k=0:n-1} c_k\mathbf{d}_k) = c_i\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i = 0.$$

As $\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i > 0$ ($\mathbf{d}_i \neq 0$ and $\mathbf{A} \succ 0$), we get $c_i = 0$. Thus $(\mathbf{d}_k)_{k=0:n-1}$ is a basis of $\mathbb{R}^n$.

2. To prove that $\mathbf{x}^*$ yields the minimum of f, note that for any $i \in 0:n-1$,

$$\begin{aligned}\mathbf{d}_i^T\nabla f(\mathbf{x}^*) &= \mathbf{d}_i^T\left(\mathbf{A}[\sum_{k=0:n-1}\frac{\mathbf{d}_k^T\mathbf{b}}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}\mathbf{d}_k] - \mathbf{b}\right)\\ &= \mathbf{d}_i^T\mathbf{A}\frac{\mathbf{d}_i^T\mathbf{b}}{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i}\mathbf{d}_i - \mathbf{d}_i^T\mathbf{b} = 0.\end{aligned}$$

Then, $\nabla f(\mathbf{x}^*) = 0$ and $\mathbf{x}^*$ yields the minimum of f.

3. To complete the proof, note that

$$\mathbf{x}_n = \mathbf{x}_0 + \sum_{k=0:n-1}(\mathbf{x}_{k+1} - \mathbf{x}_k).$$

Then, from Eq. (5) $\mathbf{x}_{k+1} - \mathbf{x}_k \propto \mathbf{d}_k$ and it comes that for any $i \in 0:n-1$,

$$\begin{aligned}\mathbf{d}_i^T\mathbf{A}\mathbf{x}_n &= \mathbf{d}_i^T\mathbf{A}(\mathbf{x}_{i+1} - \mathbf{x}_i) + \mathbf{d}_i^T\mathbf{A}\mathbf{x}_0\\ &= \frac{-1}{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i}\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i^T(\mathbf{A}\mathbf{x}_i - \mathbf{b})\mathbf{d}_i + \mathbf{d}_i^T\mathbf{A}\mathbf{x}_0\\ &= -\mathbf{d}_i^T(\mathbf{A}\mathbf{x}_i - \mathbf{b}) + \mathbf{d}_i^T A\mathbf{x}_0\\ &= -\mathbf{d}_i^T\mathbf{A}\left(\sum_{k=0:i-1}(\mathbf{x}_{k+1} - \mathbf{x}_k)\right) + \mathbf{d}_i^T\mathbf{b}\\ &= \mathbf{d}_i^T\mathbf{b}.\end{aligned}$$

Thus, for any $i \in 0:n-1$, $\mathbf{d}_i^T(\mathbf{A}\mathbf{x}_n - \mathbf{b}) = 0$ so that $\nabla f(\mathbf{x}_n) = \mathbf{A}\mathbf{x}_n - \mathbf{b} = 0$, which completes the proof. □

Algorithm 3 Conjugate gradient algorithm (for quadratic functions)

init $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{d}_0 = -\mathbf{g}_0 = -\nabla f(\mathbf{x}_0)$,
for $k = 0:n-1$ **do** ▷ (at most n loops)
  $\alpha_k = \arg\min_\alpha f(\mathbf{x}_k + \alpha\mathbf{d}_k) = -\frac{\mathbf{g}_k^T\mathbf{d}_k}{\mathbf{g}_k^T\mathbf{A}\mathbf{d}_k}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$
  $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
  $\beta_k = \frac{\mathbf{g}_{k+1}^T\mathbf{g}_{k+1}}{\mathbf{g}_k^T\mathbf{g}_k}$
  $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k$
end for
return $\mathbf{x}_n$

Now, to minimize f according to this kind of approach the problem we should be able to design conjugate directions $\mathbf{d}_{0:n-1}$. In fact, it can be shown that $\mathbf{d}_k$ can be chosen as a linear combination of gradients $(\nabla f(\mathbf{x}_i))_{i=0:k}$. Even better, starting from $\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$, for $k \geq 1$, $\mathbf{d}_k$ can be computed recursively via a linear combination of $\nabla f(\mathbf{x}_k)$ and $\mathbf{d}_{k-1}$, leading to the conjugate-gradient algorithm (algorithm 3).
Remarks
(i) Possibly, the gradient cancels before the last loop. In this case the current value of $\mathbf{x}_k$ is stuck at the optimum and the algorithm can be stopped.
(ii) Despite the fact that algorithm 3 was derived for a quadratic function, all steps of the algorithms, but the closed form for $\arg\min_\alpha f(\mathbf{x}_k + \alpha\mathbf{d}_k)$ can be evaluated for general functions, suggesting thus extension for non quadratic functions, as will be discussed at the end of the section.

Theorem 2 *When applied to function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$, if gradients $\nabla f(\mathbf{x}_k)$ do not cancel the directions $\mathbf{d}_k$ ($k = 0:n-1$) in algorithm 3 are conjugate directions. In addition, for algorithm 3, we have $\mathbf{x}_n = \mathbf{x}^*$.*

Proof letting $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$, let us note first that algorithm 3 belongs to algorithms with updates in the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=0:k} b_{ki}\mathbf{g}_i \doteq \mathbf{x}_k + \boldsymbol{\delta}_k \tag{6}$$

We are going to show that CG amounts to choose b_{ki} such that $\frac{d}{db_{ki}}f(\mathbf{x}_{k+1}) = 0$. First note that with this choice of coefficients b_{ki} corresponding gradients are orthogonal vectors. Indeed,

$$\frac{d}{db_{ki}}f(\mathbf{x}_{k+1}) = \mathbf{g}_{k+1}^T\mathbf{g}_i = 0 \text{ for } i = 0:k. \tag{7}$$

Thus, if they are nonzero the gradients form a set of orthogonal vectors. This will guarantee that the algorithm converges to the optimum $\mathbf{x}^*$ after at most n iterations.
Now, since $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$, $\mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{A}(\mathbf{x}_{k+1} - \mathbf{x}_k)$, that is,

$$\mathbf{g}_{k+1} = \mathbf{g}_k + \mathbf{A}\boldsymbol{\delta}_k. \tag{8}$$

Thus, for $i = 0:k$, orthogonality of gradients yields

$$\begin{aligned}\mathbf{g}_{k+1}^T\mathbf{g}_i &= \mathbf{g}_k^T\mathbf{g}_i + \boldsymbol{\delta}_k^T\mathbf{A}\mathbf{g}_i\\ &= \boldsymbol{\delta}_k^T\mathbf{A}\mathbf{g}_i\\ &= 0.\end{aligned}$$

But, since $\boldsymbol{\delta}_i \in span\{\mathbf{g}_j;\ j = 0:i-1\}$ for $i = 0:k-1$, $\boldsymbol{\delta}_k^T\mathbf{A}\boldsymbol{\delta}_i = 0$, that is, vectors $(\boldsymbol{\delta}_i)_{i=0,n-1}$ are conjugate w.r.t. $\mathbf{A}$.

On another hand, letting $\boldsymbol{\delta}_k = \alpha_k \mathbf{d}_k$ with

$$\mathbf{d}_k \doteq -\mathbf{g}_k + \sum_{i=0:k-1} c_{ki}\mathbf{g}_i, \tag{9}$$

relation (8) yields $\mathbf{g}_{k+1} = \mathbf{g}_k + \alpha_k \mathbf{A}\mathbf{d}_k$ and taking the dot product of both sides with $\mathbf{d}_k$ yields

$$\alpha_k = -\frac{\mathbf{g}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}.$$

Let us check that $\alpha_k = \arg\min_\alpha f(\mathbf{x}_k + \alpha \mathbf{d}_k)$:

$$\begin{aligned} \frac{d}{d\alpha} f(\mathbf{x}_k + \alpha\mathbf{d}_k) &= \mathbf{d}_k^T \nabla f(\mathbf{x}_k + \alpha \mathbf{d}_k) \\ &= (\mathbf{g}_k + \alpha \mathbf{A}\mathbf{d}_k)^T \mathbf{d}_k. \end{aligned}$$

Thus $\arg\min_\alpha f(\mathbf{x}_k + \alpha\mathbf{d}_k) = -\dfrac{\mathbf{g}_k^T\mathbf{d}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k} = \alpha_k$.
In addition, accounting for expressions

$$\begin{aligned} \mathbf{d}_{k+1} &= -\mathbf{g}_{k+1} + \textstyle\sum_{j=0:k} c_{k+1,j}\mathbf{g}_j \\ \mathbf{A}\boldsymbol{\delta}_i &= \mathbf{g}_{i+1} - \mathbf{g}_i, \end{aligned}$$

relations $\mathbf{d}_{k+1}^T \mathbf{A}\delta_i = 0$ write

$$\begin{array}{lll} i = k: & -\parallel \mathbf{g}_{k+1} \parallel^2 - c_{k+1,k} \parallel \mathbf{g}_k \parallel^2 & = 0; \\ 0 < i < k: & c_{k+1,i+1} \parallel \mathbf{g}_{i+1} \parallel^2 - c_{k+1,i} \parallel \mathbf{g}_i \parallel^2 & = 0. \end{array}$$

Putting all pieces together yields $c_{k+1,i} = -\dfrac{\parallel \mathbf{g}_{k+1} \parallel^2}{\parallel \mathbf{g}_i \parallel^2}$ for $i = 0:k$ and finally

$$\begin{aligned} \mathbf{d}_{k+1} &= -\mathbf{g}_{k+1} - \textstyle\sum_{j=0:k} \dfrac{\parallel \mathbf{g}_{k+1} \parallel^2}{\parallel \mathbf{g}_i \parallel^2}\mathbf{g}_i \\ &= -\mathbf{g}_{k+1} + \dfrac{\parallel \mathbf{g}_{k+1} \parallel^2}{\parallel \mathbf{g}_k \parallel^2}\mathbf{d}_k. \end{aligned}$$

Then, letting $\beta_k = \dfrac{\parallel \mathbf{g}_{k+1} \parallel^2}{\parallel \mathbf{g}_k \parallel^2}$ yields algorithm 3 that achieves successive minimizations of update equation (6) with respect to the coefficients b_{ki} and satisfies thus the desired properties.□

As discussed above, the use of the conjugate gradient algorithm can be extended to non quadratic functions. But in this case, one should note that in general there is no closed form formula for the optimal step α_k and approximate line search is used instead. Note also that convergence is not reached in general after n iterations. Thus, we usually restart the algorithm after n iterations with the last update of $\mathbf{x}$ as a new starting point. Finally, for non quadratic functions we get algorithm 4.
The two formulas for β_k given in algorithm 4 are equivalent in the case of quadratic functions, but this is no longer true in the general case and yield two different versions of the algorithm, known as Fletcher-Reeves and Polak-Ribière algorithms.

1.2 Second order methods

Assume a convex function $f \in C^2(\mathbb{R}^n)$ and let $\mathbf{h} = \mathbf{x} - \mathbf{x}_k$. Then

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_k) + \mathbf{h}^T \nabla f(\mathbf{x}_k) + \mathbf{h}^T \nabla^2 f(\mathbf{x}_k)\mathbf{h} + o(\parallel \mathbf{h} \parallel^2) \\ &= q_k(\mathbf{x}) + o(\parallel \mathbf{h} \parallel^2) \end{aligned} \tag{10}$$

Algorithm 4 Conjugate gradient algorithm (for non quadratic functions)

init $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{d}_0 = -\mathbf{g}_0 = -\nabla f(\mathbf{x}_0)$,
loop
  for $k = 0 : n-1$ **do**
    $\alpha_k \approx \arg\min_\alpha f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
    $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
    $\beta_k = \dfrac{\mathbf{g}_{k+1}^T(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{g}_k}$ ▷ Polak-Ribière
    or
    $\beta_k = \dfrac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$ ▷ Fletcher-Reeves
    $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$
  end for
  if Stopping condition = true **then**
    return $\mathbf{x}_n$
  else
    $\mathbf{x}_0 = \mathbf{x}_n$, $\mathbf{d}_0 = -\mathbf{g}_0 = -\nabla f(\mathbf{x}_0)$
  end if
end loop

where $q_k(\mathbf{x})$ represents the quadratic approximation of f in Taylor development about $\mathbf{x}_k$. Clearly, the minimum of q_k is obtained at

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \nabla^2 f(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k).$$

However, with this formula $\mathbf{x}_{k+1} - \mathbf{x}_k$ can be large and the approximation $f(\mathbf{x}_{k+1}) \approx q_k(\mathbf{x}_{k+1})$ may be little valid. Alternatively, the stepsize can be optimized, exactly or approximately, for instance via backtracking, in the direction $-\nabla^2 f(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$. This yields the Newton algorithm (algorithm 5):

Algorithm 5 Newton algorithm

init $\mathbf{x}_0$
while stopping condition $\neq$ true **do**
  $\mathbf{d} = -\nabla^2 f(\mathbf{x})^{-1} \nabla f(\mathbf{x})$
  $\rho \approx \arg\min_r f(\mathbf{x} + r\mathbf{d})$
  $\mathbf{x} = \mathbf{x} + \rho \mathbf{d}$
end while
return $\mathbf{x}$

Clearly, several difficulty arise with Newton algorithm. First, it is very sensitive to the convexity of f. Indeed, considering for instance $f(x) = -\frac{1}{2}x^2$, it is clear that $-\nabla^2 f(x)^{-1}\nabla f(x) = -x = \nabla f(x)$ and is not a descent direction. Another issue lies in the fact that for $f : \mathbb{R}^n \to \mathbb{R}$, as n becomes large the cost of the inversion of the Hessian matrix (generally in $O(n^3)$) makes the Newton approach unpractical. Thus, algorithms in the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{B}_k \nabla f(\mathbf{x}_k)$$

have been proposed, where $\mathbf{B}_k$ represents some approximation of the inverse of the Hessian matrix. Here $\mathbf{x}_{k+1}$ would correspond to the minimum of a second order quadratic approximation in the form $f(\mathbf{x}) \approx q_{\mathbf{B}}(\mathbf{x})$ where

$$q_{\mathbf{B}}(\mathbf{x}) = f(\mathbf{x}_k) + (\mathbf{x}_k - \mathbf{x})^T \nabla f(\mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_k - \mathbf{x})^T \mathbf{B}^{-1}(\mathbf{x}_k - \mathbf{x}).$$

Taking the gradient of this expression yields

$$\nabla q_{\mathbf{B}}(\mathbf{x}) = -\nabla f(\mathbf{x}_k) - \mathbf{B}^{-1}(\mathbf{x}_k - \mathbf{x}).$$

Thus, a convenient choice for the update $\mathbf{B}_{k+1}$ is to choose it such that $\nabla q_{\mathbf{B}_{k+1}}(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_{k+1})$, that is:

$$\mathbf{B}_{k+1}\mathbf{q}_k = \mathbf{p}_k, \tag{11}$$

where

$$\begin{array}{ll} \mathbf{p}_k & = \mathbf{x}_{k+1} - \mathbf{x}_k \\ \mathbf{q}_k & = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k). \end{array} \tag{12}$$

Eq. (11) is called the **secant equation**. The existence and interpretation of $\mathbf{B}_k = \mathbf{H}_k^{-1}$ stems from the multivariate extension of the **mean value theorem** [6] that states that for a C^1 function $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{h} \in \mathbb{R}^n$,

$$\mathbf{g}(\mathbf{x}+\mathbf{h}) - \mathbf{g}(\mathbf{x}) = \left(\int_0^1 \mathbf{g}'(\mathbf{x}+t\mathbf{h})dt \right) \mathbf{h}.$$

Then, $\left(\int_0^1 \nabla^2 f\left(\mathbf{x}_k + t(\mathbf{x}_{k+1} - \mathbf{x}_k)\right) dt \right)^{-1}$ would be a possible choice for $\mathbf{B}_{k+1}$, obtained when using $\mathbf{g} = \nabla f$. To get a simple symmetric update $\mathbf{B}_{k+1}$, Broyden proposed a rank one update:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \mathbf{v}\mathbf{v}^T.$$

Then, the secant equation writes $\mathbf{B}_k\mathbf{q}_k + \mathbf{v}\mathbf{v}^T\mathbf{q}_k = \mathbf{p}_k$, leading to

$$\mathbf{v} = \frac{1}{\mathbf{v}^T\mathbf{q}_k}\left(\mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k\right).$$

Then taking the scalar product of $\mathbf{v}$ with $\mathbf{q}_k$ also yields $(\mathbf{v}^T\mathbf{q}_k)^2 = (\mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k)^T\mathbf{q}_k$, leading finally to **Broyden's formula**:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k)(\mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k)^T}{(\mathbf{p}_k - \mathbf{B}_k\mathbf{q}_k)^T\mathbf{q}_k}. \tag{13}$$

Broyden's formula ensures symmetry of the update but, unfortunately, it does not guarantee its positivity.

To overcome the positivity issue of Broyden's formula, rank two updates have been proposed, leading to the DFP (Davidon, Fletcher and Powell) formula. DFP formula considers rank two update of the approximate Hessian matrix $\mathbf{B}_k^{-1}$. Equivalently, the BFGS (Broyden, Fletcher, Goldfarb and Shanno) formula supplies a rank two update of its inverse $\mathbf{B}_k$ that can be shown to be positive in some cases. This rank two update is obtained from the following theorem that involves a particular matrix norm, used to obtain a rank two update approximation $\mathbf{B}$ from $\mathbf{B}^*$:

Theorem 3 *Let $\mathbf{W} \succ 0$ any positive definite symmetric matrix, such that $\mathbf{W}\mathbf{p} = \mathbf{q}$ and consider the norm defined by $\| \mathbf{M} \|_{\mathbf{W}}^2 = Tr\left(\mathbf{W}\mathbf{M}^T\mathbf{W}\mathbf{M}\right) = \| \mathbf{W}^{1/2}\mathbf{M}\mathbf{W}^{1/2} \|_{\mathcal{F}}^2$, where $\| . \|_{\mathcal{F}}$ stands for the Frobenius norm. Then, the solution of the following constrained optimization problem*

$$\left\{ \begin{array}{l} \min_{\mathbf{B}} \| \mathbf{B} - \mathbf{B}^* \|_{\mathbf{W}} \\ \mathbf{B} = \mathbf{B}^T,\ \mathbf{B}\mathbf{q} = \mathbf{p} \end{array} \right.$$

is given by

$$\mathbf{B} = \left(\mathbf{I} - \rho\mathbf{p}\mathbf{q}^T\right)\mathbf{B}^*\left(\mathbf{I} - \rho\mathbf{p}\mathbf{q}^T\right)^T + \rho\mathbf{p}\mathbf{p}^T,$$

where $\rho = (\mathbf{q}^T\mathbf{p})^{-1}$.

Proof Note first that to handle the constraint $\mathbf{B} = \mathbf{B}^T$, one can either consider a term in the form $\sum_{i<j} a_{ij}(\mathbf{B}_{ij} - \mathbf{B}_{ji})$ in the Lagrangian of the problem (where the a_{ij} are Lagrange multipliers), or directly replace $\mathbf{B}_{ji}$ by $\mathbf{B}_{ij}$ for $j > i$, reducing thus the number of variables. Here we adopt the latter approach and the Lagrangian writes

$$L(\mathbf{B}) = Tr(\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{W}(\mathbf{B} - \mathbf{B}^*)) + \boldsymbol{\lambda}^T(\mathbf{B}\mathbf{q} - \mathbf{p}).$$

Letting $\boldsymbol{\Delta}_{ij}$ denote the matrix with zero entries but entry (i,j) that is equal to 1, for $i < j$ we get

$$\begin{array}{rcl} \dfrac{\partial L(\mathbf{B})}{\partial \mathbf{B}_{ij}} & = & Tr\left(\mathbf{W}(\boldsymbol{\Delta}_{ij} + \boldsymbol{\Delta}_{ji})\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\right) \\ & & +Tr\left(\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{W}(\boldsymbol{\Delta}_{ij} + \boldsymbol{\Delta}_{ji})\right) \\ & & +\boldsymbol{\lambda}^T(\boldsymbol{\Delta}_{ij} + \boldsymbol{\Delta}_{ji})\mathbf{q} \\ & = & 4\left(\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{W}\right)_{ij} + \boldsymbol{\lambda}_j\mathbf{q}_i + \boldsymbol{\lambda}_i\mathbf{q}_j \\ & = & 0. \end{array}$$

In the same way,

$$\frac{\partial L(\mathbf{B})}{\partial \mathbf{B}_{ii}} = 2\left(\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{W}\right)_{ii} + \boldsymbol{\lambda}_i\mathbf{q}_i = 0.$$

Putting all pieces together yields

$$\mathbf{C} \doteq 4\left(\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{W}\right) + \mathbf{q}\boldsymbol{\lambda}^T + \boldsymbol{\lambda}\mathbf{q}^T = 0. \tag{14}$$

Since $\mathbf{W}\mathbf{p} = \mathbf{q}$ and $\mathbf{B}\mathbf{q} = \mathbf{p}$, we have

$$\mathbf{p}^T\mathbf{C}\mathbf{p} = 4\mathbf{q}^T(\mathbf{p} - \mathbf{B}\mathbf{q}) + 2(q^T\mathbf{p})(\boldsymbol{\lambda}^T\mathbf{p}) = 0.$$

Letting $\rho = (\mathbf{q}^T\mathbf{p})^{-1}$, we get then

$$\boldsymbol{\lambda}^T\mathbf{p} = 2\rho\mathbf{q}^T(\mathbf{B}^*\mathbf{q} - \mathbf{p}).$$

and, from Eq. (14),

$$\mathbf{C}\mathbf{p} = 4\mathbf{W}(\mathbf{B} - \mathbf{B}^*)\mathbf{q} + 2\rho\mathbf{q}^T(\mathbf{B}^*\mathbf{q} - \mathbf{p})\mathbf{q} + \rho^{-1}\boldsymbol{\lambda} = 0,$$

that is,

$$\boldsymbol{\lambda} = 4\rho\mathbf{W}(\mathbf{B}^*\mathbf{q} - \mathbf{p}) - 2\rho^2\mathbf{q}\mathbf{q}^T(\mathbf{B}^*\mathbf{q} - \mathbf{p})$$

Inserting this expression of $\boldsymbol{\lambda}$ in Eq. (14) writes

$$\begin{array}{rl} \mathbf{B} & = \mathbf{B}^* - \frac{1}{4}\mathbf{W}^{-1}(\boldsymbol{\lambda}\mathbf{q}^T + \mathbf{q}\boldsymbol{\lambda}^T)\mathbf{W}^{-1} \\ & = \mathbf{B}^* - \frac{1}{4}\left[\left(4\rho(\mathbf{B}^*\mathbf{q} - \mathbf{p}) - 2\rho^2\mathbf{p}\mathbf{q}^T(\mathbf{W}\mathbf{B}^*\mathbf{q} - \mathbf{p})\right)\mathbf{p}^T \right. \\ & \quad \left. + \mathbf{p}\left(4\rho(\mathbf{q}^T\mathbf{B}^* - \mathbf{p}^T) - 2\rho^2\mathbf{q}^T(\mathbf{B}^*\mathbf{q} - \mathbf{p})\mathbf{p}^T\right)\right] \\ & = \mathbf{B}^* - \rho\mathbf{B}^*\mathbf{q}\mathbf{p}^T - \rho\mathbf{p}\mathbf{q}^T\mathbf{B}^* + \rho\mathbf{p}\mathbf{p}^T + \rho^2\mathbf{p}(\mathbf{q}^T\mathbf{B}^*\mathbf{q})\mathbf{p}^T \\ & = \left(\mathbf{I} - \rho\mathbf{p}\mathbf{q}^T\right)\mathbf{B}^*\left(\mathbf{I} - \rho\mathbf{p}\mathbf{q}^T\right)^T + \rho\mathbf{p}\mathbf{p}^T, \end{array} \tag{15}$$

which completes the proof. □

Then, the BFGS formula can be used to derive algorithm 6

Algorithm 6 BFGS algorithm

init $\mathbf{x}_0$, $\mathbf{B}_0 \succ 0$ (e.g. $\mathbf{B}_0 = \mathbf{I}$),
for $k = 0, 1, \ldots$ **do**
  $\mathbf{d}_k = -\mathbf{B}_k\nabla f(\mathbf{x}_k)$
  $\rho_k \approx \arg\min_r f(\mathbf{x}_k + r\mathbf{d}_k)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \rho_k\mathbf{d}_k$
  $\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$
  $\mathbf{q}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
  $\rho_k = (\mathbf{q}_k^T\mathbf{p}_k)^{-1}$
  $\mathbf{B}_{k+1} = \left(\mathbf{I} - \rho_k\mathbf{p}_k\mathbf{q}_k^T\right)\mathbf{B}_k\left(\mathbf{I} - \rho_k\mathbf{p}_k\mathbf{q}_k^T\right)^T + \rho_k\mathbf{p}_k\mathbf{p}_k^T$
end for

In practice, the update is generally written in the equivalent form

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k\mathbf{p}_k\mathbf{p}_k^T\mathbf{B}_k}{\mathbf{p}_k^T\mathbf{B}_k\mathbf{p}_k} + \frac{\mathbf{q}_k\mathbf{q}_k^T}{\mathbf{q}_k^T\mathbf{p}_k} \tag{16}$$

It can be shown that provided $\mathbf{B}_0 \succ 0$, all $\mathbf{B}_k$s are also positive definite:

Theorem 4 *The BFGS with optimal stepsize yields $\mathbf{B}_{k+1} \succ 0$ provided $\mathbf{B}_k \succ 0$.*

It can also be shown that if exact line search is performed and $\mathbf{B}_k$ is reset to $\mathbf{I}$ after each n iterations, BFGS boils down to Polak-Ribiere form of conjugate gradient with periodic reset. However, BFGS is superior for non exact line search.
A reduced complexity version of BFGS algorithm, known as L-BFGS, can be found in the litterature.

2 Constrained optimization

2.1 Working set methods

These methods try to solve the constrained optimization problem by moving in a fixed set of active constraints (the working set). Then, new active constraints are added to the working set when reaching feasibility boundaries, or constraints are removed from the working set when Lagrange multipliers of active inequality constraints are negative. The process is followed until convergence is achieved.

In working set methods for constrained optimization, constraints removal is based on selection of smallest Lagrange multiplier among active contraints (see algorithm 7). The reason for such a choice is based on sensitivity analysis that we briefly discuss now: let $\mathbf{x}^*(\mathbf{c})$ denote the solution of problem

$$(P_{\mathbf{c}}) \left\{ \begin{array}{l} \min_{\mathbf{x}} f(\mathbf{x}) \\ \mathbf{g}(\mathbf{x}) = \mathbf{c} \end{array} \right. \tag{17}$$

Then, we have the following result:

Theorem 5 (sensitivity analysis) *Assume* $(\mathbf{x}^*(0), \boldsymbol{\lambda})$ *satisfy necessary and sufficient second order optimality conditions for a strict local minimum of* (P_0)*. Then for* $\mathbf{c} \in \mathcal{V}_0$*, a neighborhood of* 0*, there exists a continuously varying solution* $\mathbf{x}^*(\mathbf{c})$ *that solves* $(P_{\mathbf{c}})$*. In addition,*

$$[\nabla_{\mathbf{c}} f(\mathbf{x}^*(\mathbf{c})]_{\mathbf{c}=0} = -\boldsymbol{\lambda}.$$

Proof Consider the function

$$\mathbf{k}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{c}) \doteq \begin{pmatrix} \nabla f(\mathbf{x}) + \nabla \mathbf{g}(\mathbf{x})\boldsymbol{\lambda} \\ \mathbf{g}(\mathbf{x}) - \mathbf{c} \end{pmatrix}$$

We have $\mathbf{k}(\mathbf{x}(0), \boldsymbol{\lambda}(0), 0) = 0$. In addition, second order necessary and sufficient optimality conditions yield $\nabla^2_{\mathbf{x}} L(\mathbf{x}(0)) \succ 0$ on the space tangent to constraints and regularity of $\mathbf{x}(0)$ shows that $\nabla \mathbf{g}(\mathbf{x}(0))$ is full rank. From this, it is easy to check that the partial Jacobian matrix

$$\nabla_{\mathbf{x}\boldsymbol{\lambda}} \mathbf{k}(\mathbf{x}, \mathbf{c})|_{\mathbf{x}(0)\boldsymbol{\lambda}(0)} = \begin{pmatrix} \nabla^2_{\mathbf{x}} L(\mathbf{x}(0), \boldsymbol{\lambda}) & \nabla \mathbf{g}(\mathbf{x}(0)) \\ \nabla \mathbf{g}(\mathbf{x}(0))^T & 0 \end{pmatrix}$$

can be inverted. Thus we can apply the **theorem of implicit functions** (see lecture notes on constrained optimization principles): for $\mathbf{c} \in \mathcal{V}_0$, a neighborhood of 0, there exists a continuously varying solution $\mathbf{x}^*(\mathbf{c})$ that solves $(P_{\mathbf{c}})$. As $\nabla f(\mathbf{x}(\mathbf{c})) + \nabla \mathbf{g}(\mathbf{x}(\mathbf{c}))\boldsymbol{\lambda} = 0$ and $\mathbf{g}(\mathbf{x}(\mathbf{c})) = \mathbf{c}$, the second equality yields

$$\nabla_{\mathbf{c}} \mathbf{g}(\mathbf{x}(\mathbf{c})) = \nabla_{\mathbf{c}} \mathbf{c} = \mathbf{I} = \nabla_{\mathbf{c}} \mathbf{x}(\mathbf{c})\ \nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}(c))$$

Then,

$$\begin{aligned} \nabla_{\mathbf{c}} f(\mathbf{x}(\mathbf{c}))|_{\mathbf{c}=0} &= \nabla_{\mathbf{c}} \mathbf{x}(0)\ \nabla_{\mathbf{x}} f(\mathbf{x}(0)) \\ &= -\nabla_{\mathbf{c}} \mathbf{x}(0)\, [\nabla_{\mathbf{x}} \mathbf{g}(\mathbf{x}(0))\boldsymbol{\lambda}] \\ &= -\boldsymbol{\lambda}. \end{aligned}$$

□

As a consequence, if $\mathbf{g}_i(\mathbf{x}) = 0$ and $\lambda_i < 0$, moving the ith constraint to $\mathbf{g}_i(\mathbf{x}) = c < 0$ for small c results in $f(\mathbf{x}^*(c)) - f(\mathbf{x}^*(0)) \approx -c\lambda_i < 0$. This justifies the constraints removal strategy in the working set method: after optimization of the problem in a given working set (possibly adding new constraints if getting to their boundaries), we get a point $\mathbf{x}$. Then, if there are constraints left with negative Lagrange multipliers for the solution of $\nabla f(\mathbf{x}) + \sum_i \boldsymbol{\lambda}_i \nabla \mathbf{g}_i(\mathbf{x}) = 0$ where the sum is restricted to active constraints, one or several of them are dropped. In general, the constraint that enables faster decrease of f for a small fixed decrease of it (that is, with smallest $\boldsymbol{\lambda}_i$) is dropped. This also guarantees that the objective will decrease from the previous working set and ensures thus convergence since we cannot go back to any previously visited working sets.

Projected gradient techniques where the moves are obtained by projecting the opposite gradient on the tangent space of active constraints are often considered. For NL constraints, moves are considered in the space tangent to the working surface and then projected onto it. Let us consider here the case of linear constraints for which it is possible to move inside the tangent space while staying in the constraints domain. For the problem

$$\left\{ \begin{array}{ll} \min_{\mathbf{x}} f(\mathbf{x}) & \\ \mathbf{A}_{i:}^T \mathbf{x} = b_i & i = 1 : m \\ \mathbf{A}_{j:}^T \mathbf{x} \leq b_j & j = m+1 : m+p \end{array} \right. , \tag{18}$$

where $\mathbf{A}_{i:}$ denotes line i of matrix $\mathbf{A}$, the gradient projection method leads to algorithm 7.

Algorithm 7 Gradient projection algorithm for linear constraints

1: **init** $\mathbf{x}$ feasible
2: **while** stopping condition $\neq$ true **do**
3: find active constraints for $\mathbf{x}$: $\mathbf{A}_w \mathbf{x} = \mathbf{b}_w$
4: $\mathbf{P}^{\perp}_{\mathbf{A}_w^T} = \mathbf{I} - \mathbf{A}_w^T (\mathbf{A}_w \mathbf{A}_w^T)^{-1} \mathbf{A}_w$
5: $\mathbf{d} = P^{\perp}_{\mathbf{A}_w^T} \nabla f(\mathbf{x})$
6: **if** $\mathbf{d} \neq 0$ **then**
7: $\alpha_1 = \arg\max_{\alpha} \{\mathbf{x} + \alpha \mathbf{d}$ feasible$\}$
8: $\alpha_2 = \arg\min_{\alpha} \{f(\mathbf{x} + \alpha \mathbf{d});\ 0 \leq \alpha \leq \alpha_1\}$
9: $\mathbf{x} \to \mathbf{x} + \alpha_2 \mathbf{d}$ and return to (3)
10: **else** ▷ $(\mathbf{d} = 0)$
11: $\boldsymbol{\lambda} = -(\mathbf{A}_w \mathbf{A}_w^T)^{-1} \mathbf{A}_w \nabla f(\mathbf{x})$
12: **if** $\boldsymbol{\lambda}_i \geq 0$ for $i > m$ **then** ▷ (KKT satisfied)
13: **return**
14: **else**
15: remove constraint $i = \arg\min_{j > m} \boldsymbol{\lambda}_j$
16: return to (4)
17: **end if**
18: **end if**
19: **end while**

In step 3, equation $\mathbf{A}_w \mathbf{x} = \mathbf{b}_w$ represents constraints that characterize to the current working set. As the columns of $\mathbf{A}_w^T$ form a base of the subspace orthogonal to the space tangent to the working space, $\mathbf{P}^{\perp}_{\mathbf{A}_w^T}$ is the projection on that tangent space. In step 11, the choice of $\boldsymbol{\lambda}$ yields $\nabla f(\mathbf{x}) + \mathbf{A}_w^T \boldsymbol{\lambda} = -\mathbf{d} = 0$ and $(\mathbf{x}, \boldsymbol{\lambda})$ solves CN1 optimality conditions for the working set. Step 15 follows the constraint removal strategy described above for the working set method.

Note that for linear equality constraints the problem writes

$$\left\{ \begin{array}{l} \min_{\mathbf{x}} f(\mathbf{x}) \\ \mathbf{A}\mathbf{x} = \mathbf{b} \end{array} \right. \tag{19}$$

and can be solved via Newton algorithm (algorithm 8):

Algorithm 8 Newton algorithm with equality constraints

1: **init** $\mathbf{x}$ feasible
2: **while** stopping condition $\neq$ true **do**
3: solve

$$\begin{pmatrix} \nabla^2 f(\mathbf{x}_n) & \mathbf{A} \\ \mathbf{A}^T & 0 \end{pmatrix} \times \begin{pmatrix} \mathbf{d} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla f(\mathbf{x}_n) \\ \mathbf{b} - \mathbf{A}\mathbf{x}_n \end{pmatrix}$$

4: $\rho \approx \arg\min_r f(\mathbf{x} + r\mathbf{d})$
5: $\mathbf{x} \to \mathbf{x} + \rho\mathbf{d}$
6: **end while**

In algorithm 8, the matrix equation at step 3 solves the first order optimality conditions for the local problem

$$\mathbf{d} = \arg\min_{\mathbf{h}} \begin{cases} f(\mathbf{x}_n) + \mathbf{h}^T \nabla f(\mathbf{x}_n) + \frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x}_n)\mathbf{h}, \\ \mathbf{A}(\mathbf{x}_n + \mathbf{h}) = \mathbf{b} \end{cases}$$

In practice, the initial value $\mathbf{x}_0$ needs not to be in the set of constraints, but if it is, then clearly all $\mathbf{x}_k$ will belong to it. More general convex problems in the form $\min f(\mathbf{x})$ s.t. $\mathbf{g}(\mathbf{x}) \preceq 0$, $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be addressed combining this approach and interior point methods (see next section).

2.2 Approximation methods

Letting S denote the feasibility set, approximation methods are unconstrained approximation methods that either consist in adding a strong penalty term to the objective function f outside S (**penalty methods**) or add a a penalty term defined in S that tends to infinity when moving from inside S toward its boundary (**barrier methods**).
To better understand penalty methods, let us note that solving

$$\begin{cases} \min_{\mathbf{x}} f(\mathbf{x}) \\ \mathbf{g}(\mathbf{x}) \preceq 0 \\ \mathbf{h}(\mathbf{x}) = 0 \end{cases}$$

can be rewritten in the form of an unconstrained problem

$$\min_{\mathbf{x}} f(\mathbf{x}) + \sum_i \mathcal{I}_{\mathbb{R}}(\mathbf{g}_i(\mathbf{x})) + \sum_j \mathcal{I}_0(\mathbf{h}_j(\mathbf{x})) \tag{20}$$

where $\mathcal{I}_A(u) = 0$ if $u \in A$ and $\mathcal{I}_A(u) = +\infty$ otherwise. Unfortunately, the objective in Eq. (20) is discontinuous. Penalty methods supply a relaxed version of this criterion where it is replaced by a smoother function. Typically, we replace the problem (20) by a problem

$$\min_{\mathbf{x}} f(\mathbf{x}) + tp(\mathbf{x}) \tag{21}$$

where $t > 0$ and p is a smooth function with $p(\mathbf{x}) = 0$ in S and $p(\mathbf{x}) > 0$ outside S. A possible choice is

$$p(\mathbf{x}) = \frac{1}{2}\sum_i [\max\{0, \mathbf{g}_i(\mathbf{x})\}]^2 + \frac{1}{2}\sum_j \mathbf{h}_j^2(\mathbf{x}) \tag{22}$$

As t grows to infinity we get closer to the initial problem (20). Then the idea is to solve first the smooth problem (21) approximately for a small value of t, then increase t and iterate the process using the value of $\mathbf{x}$ obtained at the end of the previous iteration as a starting point.

Algorithm 9 Penalty algorithms

1: **init** $\mathbf{x}$ and p (e.g. use (22))
2: **while** stopping condition $\neq$ true **do**
3: solve $\min_{\mathbf{x}} f(\mathbf{x}) + tp(\mathbf{x})$
4: increase t
5: **end while**

The advantage of penalty methods is that the initial guess needs not belong to S. When an initial feasible point $\mathbf{x}$ can be obtained, barrier methods can be preferred instead. When S is such that $Int(S) \neq \emptyset$, what generally occurs in the presence of inequality constraints, a barrier function ϕ is such that $Int(S) \subset \mathbf{dom}(\phi)$ and $\lim_{\mathbf{x}\to\partial S} \phi(\mathbf{x}) = +\infty$. For constraints $\mathbf{g}(\mathbf{x}) \preceq 0$, a usual choice is the log barrier function:

$$\phi(\mathbf{x}) = -\sum_i \log[-\mathbf{g}_i(\mathbf{x})]. \tag{23}$$

Then, the initial problem (20) is replaced by

$$\min_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{t}\phi(\mathbf{x}). \tag{24}$$

As t increases the effect of penalty tends to be less sensitive inside S while the barrier effect is preserved at the boundary. The principle of barrier methods is summarized in algorithm 10:

Algorithm 10 Barrier algorithms

1: **init** $\mathbf{x} \in Int(S)$ and ϕ (e.g. use (23))
2: **while** stopping condition $\neq$ true **do**
3: solve $\min_{\mathbf{x}} f(\mathbf{x}) + \frac{1}{t}\phi(\mathbf{x})$
4: increase t
5: **end while**

2.3 The log-barrier method

Now, let us consider into more details the case of log-barrier for a convex problem in the form $\min f(\mathbf{x})$ s.t. $\mathbf{g}(\mathbf{x}) \preceq 0$, $\mathbf{A}\mathbf{x} = \mathbf{b}$. This initial problem is replaced by

$$(P_t) \begin{cases} \min_{\mathbf{x}} tf(\mathbf{x}) + \varphi(\mathbf{x}) \text{ , with } \varphi(\mathbf{x}) = -\sum_i \log[-\mathbf{g}_i(\mathbf{x})] \\ \textbf{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \end{cases}$$

where $\varphi(\mathbf{x})$ is the **logarithmic barrier function**. Then, we define the **central path** as the function $t \to \mathbf{x}_t$, such that

$$\mathbf{A}\mathbf{x}_t = \mathbf{b},\ \mathbf{g}(\mathbf{x}_t) \prec 0,\ \exists \boldsymbol{\lambda}_t,\ t\nabla f(\mathbf{x}_t) + \nabla\varphi(\mathbf{x}_t) + \mathbf{A}^T\boldsymbol{\lambda}_t = 0$$

It has a nice interpretation. Indeed, here, KKT conditions yield

$$\nabla f(\mathbf{x}_t) + \boldsymbol{\mu}_t^T \nabla \mathbf{g}(\mathbf{x}_t) + \mathbf{A}^T \boldsymbol{\lambda}_t' = 0 \text{ , with } \boldsymbol{\mu}_{t,i} = (-t\mathbf{g}_i(\mathbf{x}_t))^{-1}$$

so that conditions $\boldsymbol{\mu}_i g_i(\mathbf{x}) = 0$ for (I) are replaced by $\boldsymbol{\mu}_{t,i}\mathbf{g}_i(\mathbf{x}) = -\frac{1}{t}$. Note that staying in the domain of constraints guarantees that $\boldsymbol{\mu}_{t,i} = (-t\mathbf{g}_i(\mathbf{x}_t))^{-1} > 0$. Clearly, solutions of both problem tend to become equal as $t \to \infty$.

For instance, for the problem $\min \mathbf{c}^T\mathbf{x}$ s.t. $\mathbf{F}\mathbf{x} \preceq \mathbf{d}$, centrality conditions for the logarithmic barrier write

$$t\mathbf{c} + \sum_i \frac{\mathbf{F}_{i:}^T}{\mathbf{d}_i - \mathbf{F}_{i:}\mathbf{x}_t} = t\mathbf{c} + \mathbf{F}^T\boldsymbol{\mu}_t = \mathbf{0},$$

where $\boldsymbol{\mu}_{ti} = (\mathbf{d}_i - \mathbf{F}_{i:}\mathbf{x}_t)^{-1} \succ \mathbf{0}$. Now, if functions $f, (\mathbf{g}_i)_{i=1:m}$ are convex, the problem

$$(P_t) \begin{cases} \min tf(\mathbf{x}) + \varphi(\mathbf{x}) \text{ , with } \varphi(\mathbf{x}) = -\sum_i \log[-\mathbf{g}_i(\mathbf{x})] \\ \textbf{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b} \end{cases}$$

is convex and the distance of its minimum to

$$p^* = \min\{f(\mathbf{x});\ \mathbf{g}(\mathbf{x}) \preceq 0,\ \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

is at most m/t ($\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$):

$$0 \leq f(\mathbf{x}_t) - p^* \leq \frac{m}{t}.$$

Indeed, denoting by D the constraints domain, since $\sum_i \boldsymbol{\mu}_{it}\mathbf{g}_i(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \leq 0$ for $\mathbf{x} \in D$,

$$\begin{aligned} p^* &\doteq \inf_{\mathbf{x}\in D} f(\mathbf{x}) \\ &\geq \inf_{\mathbf{x}\in D} f(\mathbf{x}) + \textstyle\sum_i \boldsymbol{\mu}_{it}\mathbf{g}_i(\mathbf{x}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &\geq f(\mathbf{x}_t) - \tfrac{m}{t}, \end{aligned}$$

because $\boldsymbol{\mu}_{it} = -1/(t\mathbf{g}_i(\mathbf{x}_t)$ and $\mathbf{A}\mathbf{x}_t - \mathbf{b} = 0$. Thus, solving the initial problem with precision ε amounts to solving $(P_{m/\varepsilon})$ and this can be done via a succession of iterations of the Newton method for linearly constrained optimization problems as described in algorithm 11.

Algorithm 11 Log-barrier algorithm

1: **init** $\mathbf{x}_0$, with $\mathbf{A}\mathbf{x}_0 = \mathbf{b}$ and $\mathbf{g}(\mathbf{x}_0) \prec \mathbf{0}$,
   $t > 0$ and $\alpha > 1$
   $\Phi(t, \mathbf{x}) = tf(\mathbf{x}) + \phi(\mathbf{x})$
2: **while** $m/t > \varepsilon$ **do**
3: solve

$$\mathbf{h}_n = \arg \begin{cases} \min_{\mathbf{h}} \ \mathbf{h}^T\nabla_{\mathbf{x}}\Phi(t, \mathbf{x}_n) + \frac{1}{2}\mathbf{h}^T\nabla^2_{\mathbf{x}}\Phi(t, \mathbf{x}_n)\mathbf{h} \\ \text{s.t. } \mathbf{A}\mathbf{h} = 0 \end{cases}$$

4: $\rho_n \approx \arg\min_r \Phi(t, \mathbf{x}_n + r\mathbf{h})$
5: $\mathbf{x}_{n+1} = \mathbf{x}_n + \rho_n\mathbf{h}$
6: $t \to \alpha t$
7: **end while**

Note that often several Newton iterations (steps 2-3), say k, are performed for each value of t. In practice $k \in [10, 20]$ works well in general. In additions, nice theoretical results for this approach can be found in [1].

2.4 Cutting hyperplane methods

In the case of problems in the form $\min f(\mathbf{x})$ s.t. $\mathbf{x} \in S$, where S is convex, possibly representing some convex approximation of an initial constraint set. For instance, S can be the convex hull of some set, that is the smallest convex set that contains it. The structure of S can be complex. However, there is a nice characterization of convex sets in terms of intersection of half spaces. More precisely, let us recall here some basic facts about convex sets:

- a polytope is characterized by a set of linear inequalities
- for a convex set S and any point $\mathbf{x} \notin S$ there exists an hyperplane that separates them
- a convex set S coincides with the intersection of half spaces limited by its supporting hyperplanes

These ideas lead to cutting hyperplane methods, the prniciple of which is given in algorithm 12:

Algorithm 12 Cutting plane algorithms

1: **init** P_0 polytope approximation of S
2: **while** stopping condition $\neq$ true **do**
3: $\mathbf{x}_k = \arg\min_{\mathbf{x}\in P_k} f(\mathbf{x})$
4: find $H_k = \{\mathbf{x};\ \mathbf{a}_k^T\mathbf{x} = b_k\}$ separating $\mathbf{x}_k$ and S
5: let $E_k = \{\mathbf{x};\ \mathbf{a}_k^T\mathbf{x} \leq b_k\}$ with $S \subset E_k$ and $\mathbf{x}_k \in E_k^c$
6: $P_{k+1} = P_k \cap E_k$
7: **end while**

For the choice of E_k, when S is defined by a set of convex inequality constraints $\mathbf{g}(\mathbf{x}) \preceq 0$, **Kelley's method** suggests to choose E_k in the following way:

$$\begin{aligned} i &= \arg\max_j \mathbf{g}_j(\mathbf{x}_k) \\ E_k &= \{\mathbf{x};\ \mathbf{g}_i(\mathbf{x}_k) + \nabla\mathbf{g}_i(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) \leq 0\}. \end{aligned}$$

Kelley's method converges to a solution of the optimization problem [2].

2.5 Duality

Let us consider the problem $\min f(\mathbf{x})$, s.t. $\mathbf{g}(\mathbf{x}) \preceq 0$, $\mathbf{h}(\mathbf{x}) = 0$. Let us denote its solution by $f(\mathbf{x}^*) = p^*$. The Lagrangian of the problem is

$$L(\mathbf{x}, \boldsymbol{\lambda}, w\boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T\mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^T\mathbf{g}(\mathbf{x})$$

Clearly, for $\boldsymbol{\mu} \succeq 0$, we have

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f(\mathbf{x}) + \mathcal{I}_{\{0\}}(\mathbf{h}(\mathbf{x})) + \mathcal{I}_{\mathbb{R}^m_-}(\mathbf{g}(\mathbf{x})).$$

Thus, $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is a continuous underestimate of the right hand term which is an objective equivalent to the initial problem. Now, let us introduce the **Dual function** defined by

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}\in\mathcal{D}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \text{ with } \mathcal{D} = dom(\mathbf{g}) \cap dom(\mathbf{h}).$$

Then, letting

$$d^* \doteq \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}\succeq 0} \phi(\boldsymbol{\lambda}, \boldsymbol{\mu}),$$

the quantity $p^* - d^*$ is called the **duality gap**.

Theorem 6 (weak duality) *The duality gap is positive:*

$$p^* - d^* \geq 0.$$

Proof or any function $k(\mathbf{w}, \mathbf{z})$ the **max-min inequality** [5] is satisfied:

$$\sup_{\mathbf{z}} \inf_{\mathbf{w}} k(\mathbf{w}, \mathbf{z}) \leq \inf_{\mathbf{w}} \sup_{\mathbf{z}} k(\mathbf{w}, \mathbf{z}).$$

The result is obtained when $k = L$, $\mathbf{w} = \mathbf{x}$ and $\mathbf{z} = (\boldsymbol{\mu}, \boldsymbol{\lambda})$. $\square$

As an example, let us consider the problem $\min \frac{1}{2} \parallel \mathbf{x} \parallel^2$, s.t. $\mathbf{C}\mathbf{x} = \mathbf{e}$. Then, $\nabla L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{x} + \mathbf{C}^T\boldsymbol{\lambda}$ and for fixed $\boldsymbol{\lambda}$ the minimum of L is reached for $\mathbf{x} = -\mathbf{C}^T\boldsymbol{\lambda}$. This leads to

$$\begin{aligned} \phi(\boldsymbol{\lambda}) &= L(-\mathbf{C}^T\boldsymbol{\lambda}, \boldsymbol{\lambda}) \\ &= (1/2)\boldsymbol{\lambda}^T\mathbf{C}\mathbf{C}^T\boldsymbol{\lambda} + \boldsymbol{\lambda}^T(-\mathbf{C}\mathbf{C}^T\boldsymbol{\lambda} - \mathbf{e}) \\ &= -(1/2)\boldsymbol{\lambda}^T\mathbf{C}\mathbf{C}^T\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{e} \end{aligned}$$

On another hand, first oder necessary optimality conditions (NC1) write

$$(\mathbf{x}^*, \boldsymbol{\lambda}^*) = (\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{e},\ -(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{e}).$$

Then,

$$\begin{aligned} d^* &= \max_{\boldsymbol{\lambda}} \phi(\boldsymbol{\lambda}) \\ &= \phi(\boldsymbol{\lambda}^*) \\ &= (1/2)\ \mathbf{e}^T(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{e} \\ &= (1/2) \parallel \mathbf{x}^* \parallel^2 \\ &= p^* \end{aligned}$$

Thus, it appears that there are cases where maximizing ϕ yields the multipliers that solve the NC1 optimality conditions and the duality gap is 0. When it is satisfied, this property is called **strong duality**. Note also that in the previous example ϕ is concave. In fact, ϕ is always concave in the set where it is finite since it represents the minimum of a set of concave functions.

Now, for simplicity, we can assume assume inequality constraints only since $\mathbf{h}(\mathbf{x}) = 0 \Leftrightarrow [\mathbf{h}(\mathbf{x}) \preceq 0 \text{ and } -\mathbf{h}(\mathbf{x}) \preceq 0]$. Note that

$$\sup_{\boldsymbol{\mu} \succeq 0} L(\mathbf{x}, \boldsymbol{\mu}) = \begin{cases} f(\mathbf{x}) \text{ if } \mathbf{g}(\mathbf{x}) \preceq 0 \\ +\infty \text{ otherwise} \end{cases}$$

Thus $p^* = \inf_{\mathbf{x}} \sup_{\boldsymbol{\mu} \succeq 0} L(\mathbf{x}, \boldsymbol{\mu})$ and weak duality writes:

$$d^* = \sup_{\boldsymbol{\mu} \succeq 0} \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\mu}) \leq \inf_{\mathbf{x}} \sup_{\boldsymbol{\mu} \succeq 0} L(\mathbf{x}, \boldsymbol{\mu}) = p^*.$$

When it holds, strong duality writes

$$d^* = \sup_{\boldsymbol{\mu} \succeq 0} \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} \sup_{\boldsymbol{\mu} \succeq 0} L(\mathbf{x}, \boldsymbol{\mu}) = p^*.$$

This leads us to the geometric interpretation of strong duality. as discussed in the proof of weak duality, for any multivariate function $k(\mathbf{w}, \mathbf{z})$ the **max-min inequality** is always satisfied:

$$\sup_{\mathbf{z}} \inf_{\mathbf{w}} k(\mathbf{w}, \mathbf{z}) \leq \inf_{\mathbf{w}} \sup_{\mathbf{z}} k(\mathbf{w}, \mathbf{z})$$

A **saddle point** for k is a point $(\tilde{\mathbf{w}}, \tilde{\mathbf{z}},)$ such that

$$\forall \mathbf{w}, \mathbf{z},\ k(\tilde{\mathbf{w}}, \mathbf{z}) \leq k(\tilde{\mathbf{w}}, \tilde{\mathbf{z}}) \leq k(\mathbf{w}, \tilde{\mathbf{z}})$$

Thus if the max-min inequality for k can be replaced by an equality, it has a saddle point. Then it appears that strong duality holds when $(\mathbf{x}^*, \boldsymbol{\mu})$ represents a saddle point of the Lagrangian.

It can be interesting to consider the dual problem when the computation of the dual function is simple. Since the objective ϕ of the dual is concave, maximizing it for $\boldsymbol{\mu} \succeq 0$ is a convex problem, even if the primal problem is not. When the dimension of $\mathbf{x}$ is large but there are few constraints, optimizing the dual can also be interesting. However, the dual optimum d^* does not always reach p^* Nevertheless, there are cases where strong duality guarantees exist. In particular, for convex functions, we have the following result:

Theorem 7 *(***Slater's conditions***) For a convex problem* $\min f(x)$*, s.t.* $\mathbf{g}(x) \preceq 0$*,* $\mathbf{h}(\mathbf{x}) = 0$*, if*

$$\{\mathbf{x}; \mathbf{g}(\mathbf{x}) \prec 0, \mathbf{h}(\mathbf{x}) = 0\} \neq \emptyset,$$

then strong duality holds: $d^* = p^*$

Considering saddle point interpretation of strong duality suggests the **Uzawa algorithm** (algorithm 13):

Algorithm 13 Uzawa algorithm

init $\mathbf{x}_0$ with $\mathbf{g}(\mathbf{x}_0) \prec 0$, $\boldsymbol{\mu}_0 \succ 0$
while stopping condition $\neq$ true **do**
$\mathbf{x}_{k+1} = \min_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\mu}_k)$
or

$$\begin{cases} \rho_k & \approx \arg\min_r L\left(\mathbf{x}_k - r\nabla_{\mathbf{x}} L(\mathbf{x}_k, \boldsymbol{\mu}_k)\right) \\ \mathbf{x}_{k+1} & = \mathbf{x}_k - \rho_k \nabla_{\mathbf{x}} L(\mathbf{x}_k, \boldsymbol{\mu}_k) \end{cases}$$

$\boldsymbol{\mu}_{k+1} = [\boldsymbol{\mu}_k + \beta_k \mathbf{g}(\mathbf{x}_k)]_+$
end while

In algorithm 13, $[\mathbf{z}]_+$ denotes the vector with i-th entry equal to $\max(\mathbf{z}_i, 0)$. Note also that a projected gradient is considered for the update of $\boldsymbol{\mu}$. Convergence guarantees are obtained when hypotheses such as strong convexity of f and convenient choice for β_k can be satisfied [3].

References

[1] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
https://web.stanford.edu/ boyd/cvxbook/

[2] D.G. Luenberger, Y. Ye, *Linear and Nonlinear Programming*, 3rd Ed., Springer, 2008.

[3] P.G. Ciarlet, *Introduction à l'Analyse Numérique Matricielle et à l'Optimisation*, Masson, 1982.

[4] M.J. Kochenderfer, T. A. Wheeler, *Algorithms for optimization*, MIT Press, 2019.

[5] Wikipedia: Max–min inequality
https://en.wikipedia.org/wiki/Max-min_inequality

[6] Wikipedia: mean value theorem
https://en.wikipedia.org/wiki/Mean_value_theorem

[7] Wikipedia: Wolfe conditions
https://en.wikipedia.org/wiki/Wolfe_ conditions