TP Projet — Documentation

Table des matières

1 Introduction

Ce document constitue la documentation nécessaire à la réalisation du TP-Projet d'Optimisation Numérique, dont les deux objectifs principaux sont les suivants :

— programmer des méthodes numériques pour la résolution de problèmes d'optimisation avec et sans contraintes ;

— réaliser une étude numérique des algorithmes implémentés.

La première partie de ce projet porte sur les problèmes d'optimisation en dimension finie sans contraintes. On étudie la méthode de Newton et sa globalisation par l'algorithme des régions de confiance. La résolution du sous-problème des régions

de confiance sera réalisée de deux façons, soit à l'aide du point de Cauchy, soit par l'algorithme du Gradient Conjugué Tronqué. La seconde partie du projet exploite la partie précédente pour résoudre des problèmes d'optimisation avec contraintes par l'algorithme du Lagrangien augmenté.

2 Consignes et date du rendu

Le non respect des consignes suivantes impliquera une note de 0.

Lorsque vous aurez terminé ce travail (**et uniquement à ce moment-là**), il vous faut

1. renommer le répertoire `projet-optinum` en `nom_prenom_groupe` ;
2. compresser ce répertoire

   commande : `tar -cvzf nom_prenom_groupe.tgz nom_prenom_groupe`

3. déposer le fichier compressé `nom_prenom_groupe.tgz` dans Moodle (dans le bon groupe) **les fichiers `.rar` sont à prohiber.**

La date limite du rendu est le vendredi 15 décembre 2023 à 18h00.

3 Optimisation sans contraintes

Dans cette partie, on s'intéresse à la résolution du problème

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

où la fonction f est au moins de classe $\mathcal{C}^2$ sur $\mathbb{R}^n$. On cherchera à exploiter l'information fournie par ses dérivées première et seconde, que l'on représente en tout point x par le *vecteur gradient* $\nabla f(x) \in \mathbb{R}^n$ et la *matrice hessienne* $\nabla^2 f(x) \in \mathbb{R}^{n\times n}$.

3.1 Méthode de Newton

Si le point $x^\star \in \mathbb{R}^n$ est solution du problème d'optimisation (1), alors il vérifie la condition nécessaire à l'ordre 1 :

$$\nabla f(x^\star) = 0_{\mathbb{R}^n}.$$

Autrement dit, toute solution au problème (1) est un proint critique de la fonction f. Une première approche pour la résolution des problèmes d'optimisation sans contraintes est donc de chercher des points critiques de la fonction f. La **méthode de Newton** va nous servir à calculer des points critiques.

3.1.1 Principe pour la résolution de systèmes d'équations non linéaires

On s'intéresse ici au point de vue résolution de systèmes non linéaires. Soit le système d'équations non linéaires suivant :

$$\varphi(x) = 0_{\mathbb{R}^n}, \quad \varphi\colon \mathbb{R}^n \to \mathbb{R}^n, \tag{2}$$

φ dérivable et non linéaire. La méthode de Newton consiste à résoudre le problème (2) de manière itérative. En entrée de la méthode, on donne la fonction φ et un itéré initial x_0. Ensuite, à chaque itération, la méthode de Newton résout une approximation linéaire du problème (2) au voisinage de l'itéré courant et met à jour l'itéré.

Soient k l'itération courante et x_k l'itéré courant. L'approximation linéaire donne :

$$\varphi(x_k + d) = \varphi(x_k) + \varphi'(x_k) \cdot d + o(\|d\|), \quad d \in \mathbb{R}^n.$$

On pose

$$\varphi_k(d) \coloneqq \varphi(x_k) + \varphi'(x_k) \cdot d$$

l'approximation linéaire. On cherche alors l'itéré suivant sous la forme :

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

tel que

$$\varphi_k(d_k) = 0.$$

Ce système étant linéaire, si $\varphi'(x_k)$ est inversible alors l'itéré suivant est donné par

$$x_{k+1} = x_k - (\varphi'(x_k))^{-1} \cdot \varphi(x_k).$$

Remarque 1. D'un point de vue numérique, pour résoudre une système linéaire $Ax = b$, il est interdit d'inverser la matrice A ! On utilisera une méthode par factorisation. [1]

3.1.2 Application à la résolution de problèmes d'optimisation

Revenons au problème d'optimisation (1). On pose $\varphi = \nabla f$. On a alors $\varphi' = \nabla^2 f$ et l'itération devient :

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

Autrement dit, à chaque itération on est amené à résoudre le sytème linéaire

$$\nabla^2 f(x_k)\, d = -\nabla f(x_k).$$

D'un point de vue de l'optimisation, nous pouvons justifier la méthode de Newton ainsi. La fonction f étant $\mathcal{C}^2$, on peut remplacer f au voisinage de l'itéré courant x_k par son développement de Taylor au second ordre, soit :

$$f(y) \sim q_k(y) \coloneqq f(x_k) + \nabla f(x_k)^T (y - x_k) + \frac{1}{2}(y - x_k)^T \nabla^2 f(x_k)\,(y - x_k).$$

On choisit alors comme point x_{k+1} le minimum de la quadratique q_k lorsqu'il existe et est unique, ce qui n'est le cas que si $\nabla^2 f(x_k)$ est définie positive. Dans ce cas, la fonctionnelle q_k est (stictement) convexe et donc le minimum de q_k est réalisé par la solution de $\nabla q_k(y) = 0$. Ainsi, l'itéré suivant x_{k+1} vérifie

$$\nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0,$$

1. https://docs.julialang.org/en/v1/stdlib/LinearAlgebra/#Standard-functions

ou encore, puisque $\nabla^2 f(x_k)$ est inversible :

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

Cette méthode est bien définie si à chaque itération, la matrice hessienne $\nabla^2 f(x_k)$ est définie positive : ceci est vrai en particulier au voisinage de la solution $x^\star$ cherchée si on suppose que $\nabla^2 f(x^\star)$ est définie positive (par continuité de $\nabla^2 f$).

Remarque 2. La méthode ne doit cependant jamais être appliquée en utilisant une inversion de la matrice hessienne (qui peut être de très grande taille et mal conditionnée), mais plutôt en utilisant :

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

où d_k est l'unique solution du système linéaire

$$\nabla^2 f(x_k)\, d = -\nabla f(x_k).$$

On appelle alors d_k la *direction de Newton.*

3.1.3 Algorithme

L'algorithme 1 est l'algorithme de Newton. Attention, la véritable spécification de la méthode du point de vue informatique est donnée dans les fichiers de code. Ici, nous ne donnons pas l'algorithme à son dernier raffinage. Nous ferons de même pour l'ensemble des algorithmes.

Algorithm 1 Résolution de $\nabla f(x) = 0$ par la méthode de Newton

input: $f \colon \mathbb{R}^n \to \mathbb{R}$, $x_0 \in \mathbb{R}^n$ (itéré initial)
output: une approximation de la solution de $\nabla f(x) = 0$

```
function Newton(f, x_0)
    k ← 0
    while Non arrêt do                                  ▷ Voir Section 3.1.4
        d_k ← solution de ∇²f(x_k) d = −∇f(x_k)          ▷ Calcul de la direction de Newton
        x_{k+1} ← x_k + d_k                             ▷ Mise à jour de l'itéré
        k ← k + 1
    end while
    return x_k
end function
```

3.1.4 Critères d'arrêt

La méthode de Newton est un processus itératif. Il nous faut donc définir des critères permettant d'arrêter le processus.

0. Le premier critère important correspond à la satisfaction à l'erreur numérique près de la condition nécessaire à l'ordre 1 : $\nabla f(x) = 0$.

Deux autres critères courants concernent la stagnation de l'algorithme. On peut considérer que si l'algorithme ne progesse plus assez, en un certain sens à préciser, que l'on veuille le stopper afin d'éviter des calculs inutiles. Ceci peut s'avérer important pour les problèmes où le coût d'une itération est très important. On considère

1. la stagnation de l'itéré ;
2. la stagnation de la valeur de la fonction f.

La méthode de Newton est un processus itératif. Ainsi,

3. un critère important est le nombre maximal d'itérations autorisées, afin d'éviter que l'algorithme ne boucle à l'infini.

La Table 1 regroupe l'ensemble des critères d'arrêts importants pour l'algorithme de Newton et les algorithmes suivants.

Remarque 3. Attention, il faut respecter l'ordre de priorité pour les critères d'arrêts. Le critère CN1 est le plus prioritaire, les itérations max le moins prioritaire. Ceci veut dire que si au cours d'une itération plusieurs critères sont vérifiés, alors, il faut que le `flag` de sortie retourne le critère le plus prioritaire.

0. CN1	$\|\nabla f(x_{k+1})\| \leq \max(\text{tol\_rel} \, \|\nabla f(x_0)\|, \text{tol\_abs})$
1. Stagnation de l'itéré	$\|x_{k+1} - x_k\| \leq \varepsilon \max(\text{tol\_rel} \, \|x_k\|, \text{tol\_abs})$
2. Stagnation de la fonction	$\|f(x_{k+1}) - f(x_k)\| \leq \varepsilon \max(\text{tol\_rel} \, \|f(x_k)\|, \text{tol\_abs})$
3. Nb d'itérations max	$k + 1 = \text{max\_iter}$

Table 1 – Les différents critères d'arrêts pour la méthode de Newton. Les critères de stagnation sont plus contraignants ($\varepsilon = 0.01$ par exemple).

3.2 Méthodes des régions de confiance

L'introduction d'une *région de confiance* dans la méthode de Newton permet de garantir la convergence globale de celle-ci, *i.e.* la convergence vers un optimum local quel que soit le point de départ. Cela suppose certaines conditions sur la résolution locale des sous-problèmes issus de la méthode, qui sont aisément imposables.

3.2.1 Principe général

L'idée de la méthode des régions de confiance est d'approcher la fonction f à l'itération k par une fonction modèle plus simple m_k dans une région

$$R_k \coloneqq \{x_k + s \mid \|s\| \leq \Delta_k\}$$

pour un Δ_k fixé. Cette région dite "de confiance" doit être suffisament petite pour que

$$m_k(s) \sim f(x_k + s)$$

si $x_k + s \in R_k$. Le principe est alors le suivant. Au lieu de résoudre le problème

$$\min_s f(x_k + s) \quad \text{t.q.} \quad \|s\| \leq \Delta_k,$$

on résout le problème supposé plus simple :

$$\min m_k(s) \quad \text{t.q.} \quad \|s\| \leq \Delta_k. \tag{3}$$

Notons s_k la solution supposée unique de (3). Si la différence entre $f(x_k + s_k)$ et $m_k(s_k)$ est trop grande, on diminue le rayon de la région de confiance Δ_k et on résout le modèle (3) à nouveau. Un avantage de cette méthode est que toutes les directions sont prises en compte. Sinon, on met à jour l'itéré :

$$x_{k+1} = x_k + s_k.$$

Exemple de modèle. On peut considérer l'approximation de Taylor à l'ordre 2 (modèle quadratique). On pose pour la suite

$$m_k(s) = q_k(s) \coloneqq f(x_k) + g_k^\top s + \frac{1}{2} s^\top H_k s \tag{4}$$

avec $g_k \coloneqq \nabla f(x_k)$ et $H_k \coloneqq \nabla^2 f(x_k)$.

3.2.2 Algorithme

Voir Algorithme 2 pour la description de l'algorithme des régions de confiance. Pour les critères d'arrêts, on fera attention à ne pas tester les stagnations si l'itéré n'est pas mis à jour. Dans l'algorithme, le calcul de s_k se fera soit par la pas de Cauchy, soit par le pas du gradient conjugué tronqué vus par la suite.

3.2.3 Le pas de Cauchy

On considère ici le modèle quadratique $q_k(s)$. Le sous-problème des régions de confiance correspondant peut se révéler difficile à résoudre (parfois autant que le problème de départ). Il est donc intéressant de se restreindre à une résolution approchée de ce sous-problème. On remarque que la direction $d_k = -g_k$ est une direction stricte de descente :

$$(\nabla f(x_k) \,|\, d_k) = -\|\nabla f(x_k)\|^2 < 0 \quad \text{si} \quad \nabla f(x_k) \neq 0.$$

Le pas de Cauchy appartient à la catégorie des solutions approchées. Il s'agit de se restreindre au sous-espace engendré par le vecteur g_k. Le sous-problème restreint s'écrit alors

$$\begin{cases} \min & q_k(s) \\ \text{t.q.} & s = -t g_k, \quad t \geq 0, \\ & \|s\| \leq \Delta_k, \end{cases} \tag{5}$$

où $\Delta_k > 0$ est le rayon de la région de confiance. Posons :

$$\varphi_k(t) \coloneqq q_k(-t g_k) = f(x_k) - t\|g_k\|^2 + \frac{1}{2} t^2 g_k^T H_k g_k =: \frac{1}{2} a t^2 + bt + c.$$

Résoudre (5) revient à résoudre le problème suivant :

$$\min \varphi_k(t) \quad \text{t.q.} \quad t \in [0\,, \frac{\Delta_k}{\|g_k\|}].$$

Algorithm 2 Résolution de $\min f(x)$, $x \in \mathbb{R}^n$, par la méthode des régions de confiance

input: $f : \mathbb{R}^n \to \mathbb{R}$, $x_0 \in \mathbb{R}^n$, $\Delta_{\max} > 0$, $\Delta_0 \in]0, \Delta_{\max}[$, $0 < \gamma_1 < 1 < \gamma_2$ et $0 < \eta_1 < \eta_2$.
output: une approximation de la solution au problème (1)

```
function RegionDeConfiance(f, x_0, Δ_max, Δ_0, γ_1, γ_2, η_1, η_2)
    k ← 0
    while Non arrêt do                                   ▷ Voir Section 3.1.4
        s_k ← solution de (3) avec Δ_k le rayon de la région de confiance
                                          ▷ Pas de Cauchy ou Gradient Conjugué Tronqué
        ρ_k ← (f(x_k) − f(x_k + s_k)) / (m_k(0_{R^n}) − m_k(s_k))
                                          ▷ Voir (4) pour le choix du modèle m_k
        if ρ_k ≥ η_1 then
            x_{k+1} ← x_k + s_k                          ▷ Mise à jour de l'itéré
        end if
        if ρ_k ≥ η_2 then
            Δ_{k+1} ← min{γ_2 Δ_k, Δ_max}                ▷ On augmente la région de confiance
        else if ρ_k ≥ η_1 then
            Δ_{k+1} ← Δ_k
        else
            Δ_{k+1} ← γ_1 Δ_k                            ▷ On diminue la région de confiance
        end if
        k ← k + 1
    end while
    return x_k
end function
```

Remarque 4. Puisque l'on se place le long d'une direction stricte de descente, on a :

$$b = -\|g_k\|^2 < 0$$

si $g_k \neq 0$.

3.2.4 Le gradient conjugué tronqué

Les expérimentations numériques montrent que la technique du pas de Cauchy ne garantit pas une convergence rapide en général ; on retrouve ici le problème d'une méthode de descente de gradient. On souhaite donc étudier une méthode pour la résolution approchée du sous-problème avec région de confiance (3), qui puisse récupérer asymptotiquement la convergence quadratique inhérente à la méthode de Newton. L'algorithme du Gradient Conjugué Tronqué appartient à cette catégorie.

Voir Algorithme (3) pour la description de l'algorithme du gradient conjugué tronqué.

Algorithm 3 Résolution de (3) par la méthode du gradient conjugué tronqué

input: $\Delta > 0$, $x \in \mathbb{R}^n$, $g = \nabla f(x) \in \mathbb{R}^n$, $H = \nabla^2 f(x) \in \mathbb{R}^{n \times n}$.
output: une approximation de la solution au sous-problème $\min q(s) \coloneqq g^T s + \frac{1}{2} s^T H s$, $\|s\| \leq \Delta$

function GCT(g, H, Δ)
  $j \leftarrow 0$
  $g_0 \leftarrow g$
  $s_0 \leftarrow 0_{\mathbb{R}^n}$
  $p_0 \leftarrow -g$
  while $j < 2n$ et $\|g_j\| > \max(\|g_0\| \, \text{tol\_rel}, \text{tol\_abs})$ **do** ▷ Voir Section 3.1.4
    $\kappa_j \leftarrow p_j^T H p_j$
    if $\kappa_j \leq 0$ **then**
      $\sigma_j \leftarrow$ la racine de $\|s_j + \sigma p_j\| = \Delta$ pour laquelle $q(s_j + \sigma p_j)$ est la plus petite
      return $s_j + \sigma_j p_j$
    end if
    $\alpha_j \leftarrow {g_j}^T g_j / \kappa_j$
    if $\|s_j + \alpha_j p_j\| \geq \Delta$ **then**
      $\sigma_j \leftarrow$ la racine positive de $\|s_j + \sigma p_j\| = \Delta$
      return $s_j + \sigma_j p_j$
    end if
    $s_{j+1} \leftarrow s_j + \alpha_j p_j$
    $g_{j+1} \leftarrow g_j + \alpha_j H p_j$
    $\beta_j \leftarrow \dfrac{{g_{j+1}}^T g_{j+1}}{g_j^T g_j}$
    $p_{j+1} \leftarrow -g_{j+1} + \beta_j p_j$
    $j \leftarrow j + 1$
  end while
  return s_j
end function

4 Optimisation avec contraintes — Lagrangien augmenté

Dans cette partie, nous nous intéressons à la résolution des problèmes d'optimisation sous contraintes. Le problème se présente donc sous la forme :

$$\min\ f(x) \quad \text{sous la contrainte} \quad x \in C \subset \mathbb{R}^n,$$

où C est un sous-ensemble non vide fermé de $\mathbb{R}^n$.

4.1 Principe de la méthode du lagrangien augmenté

La méthode du lagrangien augmenté appartient à une classe d'algorithmes qui permettent la résolution des problèmes avec contraintes. Elle s'apparente aux méthodes de pénalisation, dans lesquelles on résout le problème avec contraintes à travers une suite de problèmes sans contraintes.

4.2 Algorithme dans le cas de contraintes d'égalité

On s'intéresse ici au cas où l'ensemble C est défini par un ensemble des contraintes d'égalités. Le problème se met ainsi sous la forme :

$$\min f(x) \quad \text{sous la contrainte} \quad c(x) = 0_{\mathbb{R}^m} \tag{6}$$

où $c\colon \mathbb{R}^n \to \mathbb{R}^m$ est supposée suffisamment différentiable. L'algorithme 4 du lagrangien augmenté est obtenu de Bierlaire, *Introduction à l'optimisation différentiable*.

Remarque 5. Attention à bien adapter les critères d'arrêts ! Le critère d'arrêt correspondant à la condition nécessaire à l'ordre 1 fait intervenir le lagrangien (non augmenté) associé au problème (6) et la contrainte c.

Pour la résolution du sous-problème sans contraintes, on pourra utiliser une méthode des régions de confiance ou un algorithme de Newton par exemple.

Algorithm 4 Résolution de $\min f(x)$, $c(x) = 0_{\mathbb{R}^m}$, par la méthode du lagrangien augmenté

input: $f\colon \mathbb{R}^n \to \mathbb{R}$, $c\colon \mathbb{R}^n \to \mathbb{R}^m$, $x_0 \in \mathbb{R}^n$, $\lambda_0 \in \mathbb{R}^m$, $\mu_0 > 0$, $\tau > 0$
output: une approximation de la solution au problème (6)

function LagrangienAugmente(f, c, x_0, λ_0, μ_0, τ)
$k \leftarrow 0$
$\beta \leftarrow 0.9$
$\hat{\eta} \leftarrow 0.1258925$
$\alpha \leftarrow 0.1$
$\varepsilon_0 \leftarrow 1/\mu_0$
$\eta_0 \leftarrow \hat{\eta}/\mu_0^\alpha$
while Non arrêt **do** ▷ Voir Section 3.1.4. Attention, il faut adapter.
▷ Ne pas mettre de critères de stagnation.
$x_{k+1} \leftarrow$ un minimiseur du problème sans contraintes

$$\min_{x \in \mathbb{R}^n} L_A(x, \lambda_k, \mu_k) \coloneqq f(x) + \lambda_k^T c(x) + \frac{\mu_k}{2}\|c(x)\|^2$$

avec x_k comme point de départ, en terminant lorsque $\|\nabla_x L_A(\cdot, \lambda_k, \mu_k)\| \leq \varepsilon_k$.
if $\|c(x_{k+1})\| \leq \eta_k$ **then** ▷ Mettre à jour (entre autres) les multiplicateurs
$\lambda_{k+1} \leftarrow \lambda_k + \mu_k c(x_{k+1})$
$\mu_{k+1} \leftarrow \mu_k$
$\varepsilon_{k+1} \leftarrow \varepsilon_k/\mu_k$
$\eta_{k+1} \leftarrow \eta_k/\mu_k^\beta$
else ▷ Autrement, mettre à jour (entre autres) le paramètre de pénalité
$\lambda_{k+1} \leftarrow \lambda_k$
$\mu_{k+1} \leftarrow \tau\mu_k$
$\varepsilon_{k+1} \leftarrow \varepsilon_0/\mu_{k+1}$
$\eta_{k+1} \leftarrow \hat{\eta}/\mu_{k+1}^\alpha$
end if
$k \leftarrow k + 1$
end while
return x_k, λ_k, μ_k
end function