TP projet : Optimisation numérique

ATTACHE William

semestre 5, 2017

Table des matières

Chapitre 1

Introduction

Le but de ce projet est d'étudier la résolution de problèmes d'optimisation avec et sans contrainte. En première partie, on s'intéressera aux problèmes de minimisation sans contrainte, d'abord grâce à l'algortihme de Newton puis grâce à sa globalisation par la méthode des régions de confiance. La résolution des sous-problèmes dans l'algorithme des régions de confiance sera faite soit par l'intermédiaire du point de Cauchy, soit par le biais de l'algorithme de Moré-Sorensen. Dans un deuxième temps, on s'intéressera à la méthode du Lagrangien augmenté, pour résoudre des problèmes d'optimisation avec contraintes.

Remarques préliminaires :

— dans les contrats des fonctions Matlab, on parlera de "fonction gradient" et "fonction hessienne" pour parler respectivement de $\nabla f : \mathbb{R}^n \mapsto \mathcal{L}(\mathbb{R}, \mathbb{R})$ et $Hf : \mathbb{R}^n \mapsto \mathcal{L}(\mathbb{R}^n, \mathbb{R})$.

Chapitre 2

Problèmes d'optimisation sans contrainte : l'algortihme de Newton

2.1 Algortihme de Newton local

On présente l'algorithme de manière générale.

Algorithm 1 Calcul approché de $\mathcal{S}$

Data: f,x0

Result: Algorithme de Newton

`/* Coefficient d'approximation */`

$\epsilon \leftarrow 10^{-9}$ `/* Coefficient des garde-fou */`

$\kappa \leftarrow 10^{3}$ `/* Calcul de la valeur approchée */`

$k \leftarrow 1$ $\gamma \leftarrow \gamma_1$ $\mathcal{S} \leftarrow 0$ **while** $(\gamma \geq \epsilon) \wedge (k \leq \kappa)$ **do**

  $\mathcal{S} \leftarrow \mathcal{S} + \gamma$ $k \leftarrow k + 1$ $\gamma \leftarrow \gamma_k$

end

Pour trouver d_k, comme on résout le système $\nabla^2 f(x) * d_k = -\nabla f(x)$, on n'a pas besoin de supposer $\nabla^2 f(x)$ inversible. cet algorithme est donc efficace pour trouver une solution au problème de minimi-

sation sans contrainte au sens où l'on ne contraint pas beaucoup $\nabla^2 f(x)$, notamment on ne lui impose pas d'être inversible.

En revanche, la convergence de cet algorithme dépend du point de départ, ce qui constitue sa principale faiblesse. Rappelons qu'analytiquement, la condition de convergence locale de l'algorithme de Newton s'écrit : $\epsilon > 0, \|x_0 - x\star\| \leq \epsilon \Rightarrow x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$. Nous verrons comment pallier à ceci lorsque nous mettrons en place la méthode des régions de confiance.

2.2 Les critètes d'arrêt

On met en place trois critères d'arrêt pour l'algorithme de Newton, qui seront utilisés dans tous les algorithmes étudiés ci-après. On note f la fonction à minimiser.

— $\|\nabla f(x_k)\| \leq tol1 * \|\nabla f(x_0) + c\|$: grâce à ce critère, on s'arrête si les pentes changent très peu par rapport à la pente que l'on avait au point initial. On peut ainsi choisir de s'arrêter plus ou moins près du minimum, ce qui est intéressant par exemple dans un algorithme de régularisation dit "early stopping", utilisé par exemple en intelligence artificielle pour éviter le surajustement.

— nbIterations ¿ nbIterationsMax : critère simple qui sert à ne pas effectuer trop d'itérations, et à détecter parfois une erreur dans l'algortihme. Ceci a été mon cas pour l'algorithme des régions de confiance. J'arrivais à 100 000 itérations sans avoir convergé, ce qui m'a laissé entrevoir une erreur. En fait, ma première itération n'était pas bien gérée.

— $\|x_k - x_{k-1}\| \leq \|x_{k-1}$: ce critère permet de stopper l'algorithme si d'une itération à l'autre, on a "peu" bougé en allant de x_k à x_{k-1}.

2.3 Tests

On considère ici les deux fonctions vues en TD :

$$f_1 : \begin{array}{ccc} \mathbb{R}^3 & \to & \mathbb{R} \\ a \times b \times c & \mapsto & 2(a+b+c-3)^2 + (a-b)^2 + (b-c)^2 \end{array}$$

et

$$f_2 : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R} \\ a \times b & \mapsto & 100(b-a^2)^2 + (1-a)^2 \end{array}$$

Ces deux fonctions, polynômiales, sont de classe $\mathcal{C}^\infty$ sur leur domaine de définition. On peut donc calculer leurs graidents et hessiennes respectives. En outre, les deux problèmes de minimisation associés admettent une solution sur le domaine de définition de leurs fonctions respectives.

2.3.1 Tests sur f_1

On test l'algorithme sur la fonction f_1, avec les deux points de départ $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ et $\begin{pmatrix} 10 \\ 3 \\ -2.2 \end{pmatrix}$. Ces deux points assurent la convergence de l'algorithme en une seule itération.

2.3.2 Tests sur f_2

On test l'algorithme sur la fonction f_2, avec les deux points de départ $\begin{pmatrix} -1.2 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 10 \\ 0 \end{pmatrix}$, et $\begin{pmatrix} 0 \\ \frac{1}{200} + \frac{1}{10^{12}} \end{pmatrix}$.

Les deux premiers points donnent satisfaction, et trouve le point où f_2 atteint son minimum, qui est $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. En revanche, le troisième point de départ ne permet pas de trouver le point où f_2 atteint son minimum. On voit ici la faille principale de l'algorithme de Newton qui est que la convergence de l'algorithme dépend du point de départ choisi. L'algorithme renvoie $\begin{pmatrix} 0 \\ 2.1683 \end{pmatrix}$ comme solution. Si

on ne connait pas la solution au problème de minimisation en avance, on peut se douter que le point renvoyé n'est pas solution car :

— 10 itérations sont réalisées par l'algorithme avant de donner le résultat

— à chaque itération, matlab nous indique que la matrice $\nabla^2 f(x)$ est mal conditionnée ou bien quasiment singulière.

2.3.3 Interprétation :

— La fonction f_1 étant une forme quadratique, elle coïncide avec son développement de Taylor à l'ordre 2. L'algorithme de Newton implémenté converge donc en une seule itéaration pour cette fonction.

— La fonction f_2 en revanche ne donne pas satisfaction si le point de départ est le point $\begin{pmatrix} 0 \\ \frac{1}{200} + \frac{1}{10^{12}} \end{pmatrix}$. Graphiquement, nous pouvons observer ce qui se passe avec les tangentes, Figure 2.1. Celles aux points A (point de départ) et E presques confondues, on peut se dire que l'on va osciller de part et d'autre du minimum sans tomber dessus, sauf si l'on réalise peut-être un très très grand nombre d'itérations. Analytiquement, on peut regarder la hessienne de f_2 : $Hf_2((x_1, x_2)) = \begin{pmatrix} 1200 * x_1^2 - 400 * x_2 + 2 & -400 * x_1 \\ -400 * x_1 & 200 \end{pmatrix}$. Le terme $1200 * x_1^2$ nous permet de dire que l'algorithme sera très sensible au point de départ pour la fonction f_2 puisqu'il interviendra de manière forte dans sa convexité.

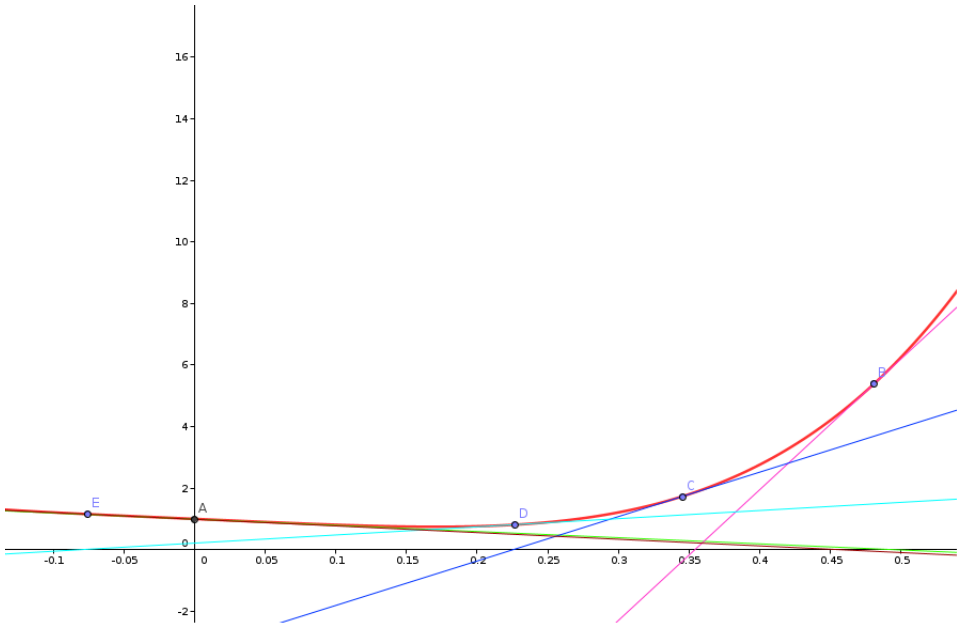

FIGURE 2.1 – Fonction f_2, avec la seconde variable fixée à $\frac{1}{200} + \frac{1}{10^{12}}$, sur laquelle on a appliqué 5 itérations de l'algorithme de Newton avec le point de départ A(0,1).

Chapitre 3

La méthode de régions de confiance : Newton devient robuste

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 quelque soit le point de départ. On se situe toujours ici dans le cas de problèmes d'optimisation sans contrainte.

3.1 Cauchy : chercher le minimum de f dans le direction opposée au gradient

L'idée est de chercher, à chaque itération, à minimiser f dans la direction opposée au gradient, à la manière de l'algorithme de la Steepest Descent. Cependant, on impose au point trouvé de rester dans un disque de rayon Δ_k centré sur le point de départ x_k de la $k^{ième}$ itération.
Le développement de Taylor de la fonction f à l'ordre 2 s'écrit ici :

$$m_k(x_k + s) = f(x_k + s) = f(x_k) + g_k^T s + \frac{1}{2} s^T H_k s$$

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

Etant donné x_k, on cherche à résoudre le problème suivant :

$$\begin{cases} min & m_k(xk+s) \\ s.t. & s = -tg_k \\ & t > 0 \\ & ||s|| \leq \Delta_k \end{cases}$$

Pour tout $k \in \mathbb{N}$, on pose $\rho_k = \dfrac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$.

Le principe est de chercher le minimum atteint par la fonction, dans le disque de rayon Δ_k, dans la direction opposée du gradient. Une fois le point s où f atteint son minimum dans cette direction trouvé, on regarde la décroissance de f entre x_k et $x_k + s$. Puis, suivant la valeur de cette décroissant par rapport à la décroissance de notre modèle entre ces deux mêmes points (ρ_k) ,on modifie ou non le rayon de la région de confiance, on modifie ou non x_k, suivant les règles suivantes :

$$x_{k+1} = \begin{cases} x_k + s & si \quad \rho_k \geq \eta_1 \\ x_k & sinon \end{cases}$$

et

$$\Delta_{k+1} = \begin{cases} min\{\gamma_2 \Delta_k, \Delta_{max}\} & si & \rho_k \geq \eta_2 \\ \Delta_k & si & \rho_k \in [\eta_1, \eta_2\,[\\ \gamma_1 \Delta_k & sinon & \end{cases}$$

Interprétation graphique : Il est possible de visulaiser graphiquement ce qui se passe lorsque l'on cherche le point de Cauchy. On note $t_{SD} = \frac{\|g_k\|^2}{g_k^T H_k g_k}$ le point donné par l'algorithme de la Steepest Descent minimisant la fonction, et $limit = \frac{\Delta_k}{\|g_k\|}$ qui sort directement de la contrainte $\|tg_k\| \leq \Delta_k$ avec $t > 0$.

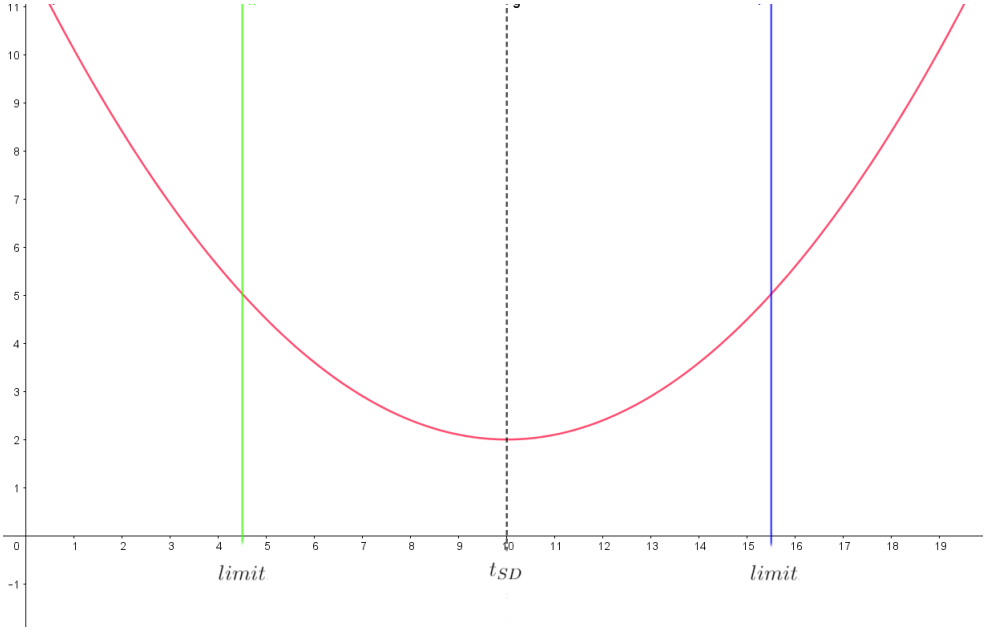

FIGURE 3.1 – Cas où la courbure de la fonction est positive. Si $t_{SD} < limit$ (cas bleu) alors le minimum est t_{SD}, sinon t_{SD} est hors des limites et dans ce cas le minimum est $limit$ (cas vert) .

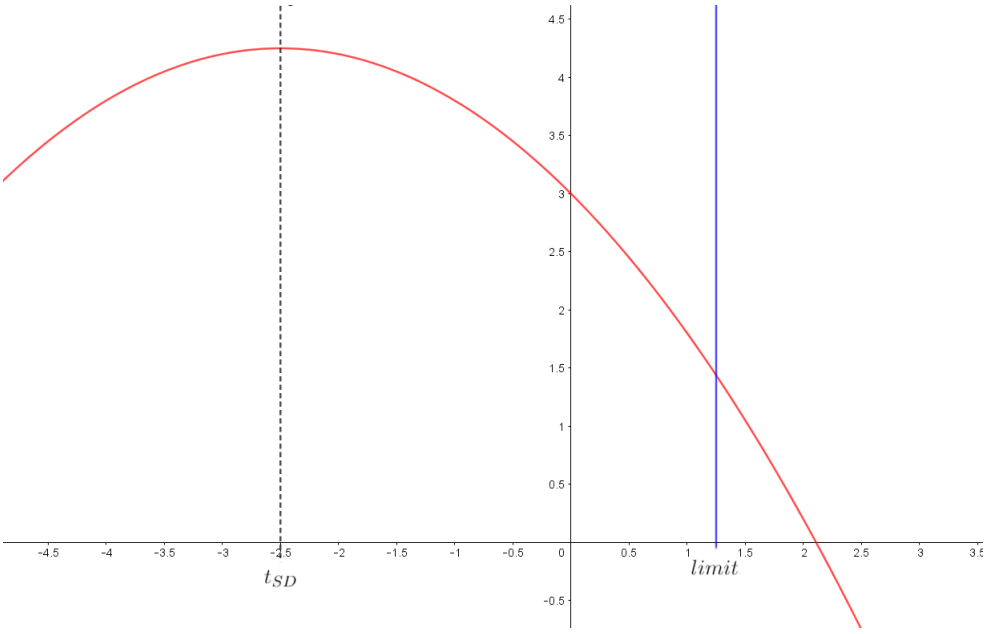

FIGURE 3.2 – Cas où la courbure est négative. Dans ce cas t_{SD} est également négatif et ne peut être retenu, le minimum est $limit$.

3.1.1 Tests de l'algorithme trouvant le point de Cauchy

On note respectivement q_1, q_2 et q_3 les quadratiques 1, 2 et 3 de l'annexe B du document du projet.

Voici la suite des tests réalisés acocmpagnés chacun d'un commentaire (le premier argument de la fonction Cauchy est le rayon de la région de confiance) :

— Cauchy(1,q_1) renvoie NaN, ce qui est normal puisque le gradient associé est $0_{\mathbb{R}^2}$. On voit ici apparaître la nécessité de prendre ce cas en compte dans l'algorithme des régions de confiance.

— Cauchy(1,q_2) $\Rightarrow$ la courbure est positive, et $t_{SD} = 0,0243 < 0,1581 = limit$. L'algorithme renvoie $t = 0,0243 = t_{SD}$, ce qui est conforme à ce que l'on attend.

— Cauchy(10^{-2},q_2) $\Rightarrow$ la courbure est positive, et $t_{SD} = 0,0243 > 0,0016 = limit$. L'algorithme renvoie $t = 0,0016 = limit$, ce qui est conforme à ce que l'on attend.

— Cauchy(1,q_3) $\Rightarrow$ la courbure est positive, et $t_{SD} = 1,1120 > 0,4472 = limit$. L'algorithme renvoie $t = 0,4472 = limit$, ce qui est conforme à ce que l'on attend.

— Cauchy(10,q_3) $\Rightarrow$ la courbure est positive, et $t_{SD} = 1,1120 < 4,4721 = limit$. L'algorithme renvoie $t = 1,1120 = t_{SD}$, ce qui est conforme à ce que l'on attend.

3.1.2 Newton et la méthode des régions de confiance

Il sagit désormais d'inclure dans l'agorithme de Newton la méthode des régions de confiance, qui permet de rendre la convergence de l'algortihme de Newton indépendante du point de départ. Cet algorithme découle de ce qui a été dit plus haut concernant la point de Cauchy et son utilisation de posteriori : à chaque itération de l'algortihme de Newton, on calcule le pas de Cauchy, puis on met à jour le rayon de la région de confiance ainsi que le point x_k où s'effectuera la prochain itération.

Tests : Cette fois, tous les tests donnent satisfaction, on trouve bien l'élément unité des espaces dans lesquels on travaille comme point minimisant la fonction f.

3.2 More-Sorensen : une convergence plus rapide d'avec Cauchy

La méthode du pas de Cauchy s'apparente donc assez à celle de la steepest descent, n'offre donc pas non plus une convergence très rapide. Pour pallier à ceci, nous étudions l'algorithme de More-Sorensen.

3.2.1 Newton pour les équations non linéaires

Un travail préliminaire à effectuer pour pouvoir implémenter l'algortihme de More-Sorensen est d'implémenter celui de Newton permettant de résoudre les équations de la forme $\phi(x) = 0$ où ϕ est une fonction non linéaire de la variable $x \in \mathbf{R}$. Si l'itération de Newton (issue du développement limité de la fonction ϕ au point où elle atteint son minimum) n'est pas acceptée, on réalise une dichotomie pour lMin et lMax pour assurer la convergence de l'algorithme. Dans cet algorithme, nous verrons apparaître des critères d'arrêt que nous expliquons ici :

— $min(|\phi(lMin)|, |\phi(lMax)|) < epsilon$: si l'un des deux bornes est près du réel minimisant f à la précision espilon alors on s'arrête

— $|\phi(lambda)| < epsilon$: si lambda lui-même (dotn on dispose sans avoir besoin de le calculer à ce moment là) est suffisament proche du réel minimisant f, on s'arrête.

Remarque : On peut se poser la question ici de savoir comment rechercher les valeur lMin et lMax si elles ne sont pas fournies, ce qui sera le cas lorsque l'on implantera More-Sorensen. Pour résoudre ce problème, on se place donc dans le cas pratique qui nous intéresse, et où la fonction ϕ est donc décroissante sur l'intervvalle où l'on recherche une solution à l'équation. On note λ_1 la plus grande valeur propre de la hessienne de la fonction f à minimiser. Alors, on cherchera systématiquement un zéro avec la méthode de Newtion qui soit sur la demi-droite $[max(0, -\lambda_1)\ ,\ +\infty\ [$. On prendra ainsi $lMin = max(0, -\lambda_1)$ (et alors $\phi(lMin) > 0$ et on cherchera lMax en incrémentant lMin d'une valeur a (choisie à .. dans mon cas) jusqu'à ce que l'on ai $\phi(lMax) < 0$.
L'algorithme de recherche du zéro est le suivant :

Algorithm 2 Newton non linéaire

Data: lMin , lMax , ϕ , $\partial\phi$

Result: lambda = zero de la fonction ϕ à la précision epsilon

$espilon = 10^{-4}$

if *lMin ou lMax est déjà le minimul à la précision epsilon* **then**
    lambda = (lMin,lMax)
else
    lambda = lMax
end
while *non convergence* **do**
    Calculer l'itéré de Newton lN
    if *lMin* $\leq$ *lN* $\leq lMax$ *et* $\|\phi(lN)\| < 0.5 * \|\|phi(lambda)\|$ **then**
        lambda = lN
    else
        procéder par dichotomie entre lMin et lMax pour trouver lambda
    end
end

Concernant les tests, tous ceux demandés ont été réalisés et donnent des résultats identiques à ceux de l'enseignant de TP, ce qui donne confiance en l'impémentation réalisée.

3.2.2 Algorithme de More-Sorensen

L'algorithme de More-Sorensen se base sur le résultat suivant :

Si s^* est une solution du problème de minimisation $\min_{\|s\| \leq \Delta} m(x+s)$ avec m : $\begin{array}{ccc} E & \rightarrow & \mathbb{R} \\ z & \mapsto & g^T z + \frac{1}{2} z^T H z \end{array}$

alors il existe λ^* vérifiant : $\begin{cases} (H+\lambda^* I)s^* &= -g \ (1) \\ \lambda^*(\|s^*\|-\Delta) &= 0 \ (2) \\ H+\lambda^* I &\geq 0 \ (3) \\ \lambda^* &\geq 0 \ (4) \\ \|s^*\| &\leq \Delta \ (5) \end{cases}$

Alors, connaissant λ^* on peut, via la relation (1), retrouver s^* grâce à la commande matlab $s = (H+\lambda^* I)\backslash -g$ et ainsi intégrer cet algorithme à celui de Newton avec régions de confiance pour trouver s (au lieu d'utiliser le pas de Cauchy).

Interprétation :

— Comparaison avec la décroissance obtenue par le pas de Cauchy :

— Avantages et inconvénients des deux approches :

	$Pas\ de\ Cauchy$	$Pas\ de\ More-Sorensen$
$Avantages$	$\cdot Simple\ à\ mettre\ en\ oeuvre$	$\cdot Convergence\ rapide^{(*)}$
$Inconvénients$	$\cdot Convergence\ lente$ $\cdot Très\ dépendent\ du\ modèle\ m$ $\Rightarrow erreur\ d'approximation$ $a\ priori\ non\ contrôlée$	$\cdot Compliqué\ à\ mettre\ en\ oeuvre$ $Nécessite\ la\ résoltuon\ d'une\ équation$ $matricielle$

$^{(*)}$: par convergence rapide, on entend en un nombre faible d'itérations (comparé à la méthode du pas de Cauchy).

Chapitre 4

Optimisation sous contraintes : la méthode du Lagrangien augmenté

L'objectif est ici de résoudre des problèmes avec contraintes, d'égalité seulement dans un premier temps. La méthode est Lagrangien augmenté est une méthode de pénalisation, où l'on se ramène à chaque itération à la résolution de problèmes sans contrainte grâce à l'introduction justement du Lagrangien de la fonction f à minimiser. Nous nous intéressons donc aux problèmes de la forme :

$$\begin{cases} min & f(x) \\ s.t. & x \in \mathbb{R}^n \\ & c(x) = 0, \;\; c : \mathbb{R}^n \mapsto \mathbb{R}^m \end{cases}$$

On donne ici le pseudo-code de l'algorithme du Lagrangien augmenté afin de commenter l'aspcet pénalisation. On remarquera que comme annoncé plus haut, les critètres d'arrêt (ou de converge) associés sous les mêmes que ceux utilisés pour l'algorithme de Newton mais associés au Lagrangien de f définit par :

$$L_A : \begin{array}{ccc} \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} & \to & \mathbb{R} \\ (x, \lambda, \mu) & \mapsto & f(x) + \lambda^T c(x) + \frac{\mu}{2}\|c(x)\|^2 \end{array}$$

L'algorithme est le suivant :

Algorithm 3 Algorithme du Lagrangien augmenté

Data: μ_0 , τ , α , β , η_0^c , ϵ_0 , η_0 et le point de départ (x_0, λ_0)

Result: une approximation de la solution du problème avec contraintes d'égalité

while *non convergence* **do**

  Calculer, à la précision $\nabla L_A(., \lambda_k, \mu_k) \leq \epsilon_k$, une solution du problème sans contrainte $\min_{x \in \mathbb{R}^n} L_A(x, \lambda_k, \mu_k)$

  if $\|c(x_{k+1})\| \leq \eta_k$ **then**

    mettre à jour le multiplicateur $\lambda_{k+1} = \lambda_k + \mu_k c(x_{k+1})$ (et autres constantes)

  else

    mettre à jour la pénalité $\mu_{k+1} = \tau \mu_k$

  end

end

On voit ainsi comment les contraintes sont prises en compte : comme $\tau > 0$, si l'on a pas $\|c(x_{k+1})\| \leq \eta_k$, on augmente μ_k de sorte à ce qu'à l'itération suivante, les contraintes aient plus de poids dans la fonction à minimiser et soient donc plus prises en compte. Sinon, on se dit que le vecteur trouvé vérifie suffisamment bien les contraintes d'égalité et donc on continue à construire la suite λ_k convergeant vers le multiplicateur de Lagrange $\lambda^\star$ associé au problème.