© *J.-B.A.K. <jean-baptiste.apoung@math.u-psud.fr>*

TP4 : Optimisation sous contrainte en dimension finie : reconstruction d'un signal par projection sur les fonctions convexes.

Note 1 (Attention)**.**

Dans cette fiche nous regarderons un problème d'optimisation sous contraintes d'inégalité pour lequel il n'est pas aisé de construire l'opérateur de projection sur l'espace des contraintes.
La fiche est constituée de deux parties :

- Un travail effectif à réaliser pendant la séance de TP.
- Un travail à rendre sous forme d'un rapport et d'une présentation pour le **24 Janvier 2018**.

Le travail à rendre concernera entre autres :

- **les réponses aux questions théoriques posées dans l'énoncé Parie-1**
  (*proposées par le chargé du cours pour juger de la compréhension des techniques vues en cours.*)
- **la mise en oeuvre de la méthode de pénalisation Partie-2-4**

Partie - 1 *Le problème modèle et sa discrétisation par éléments finis*

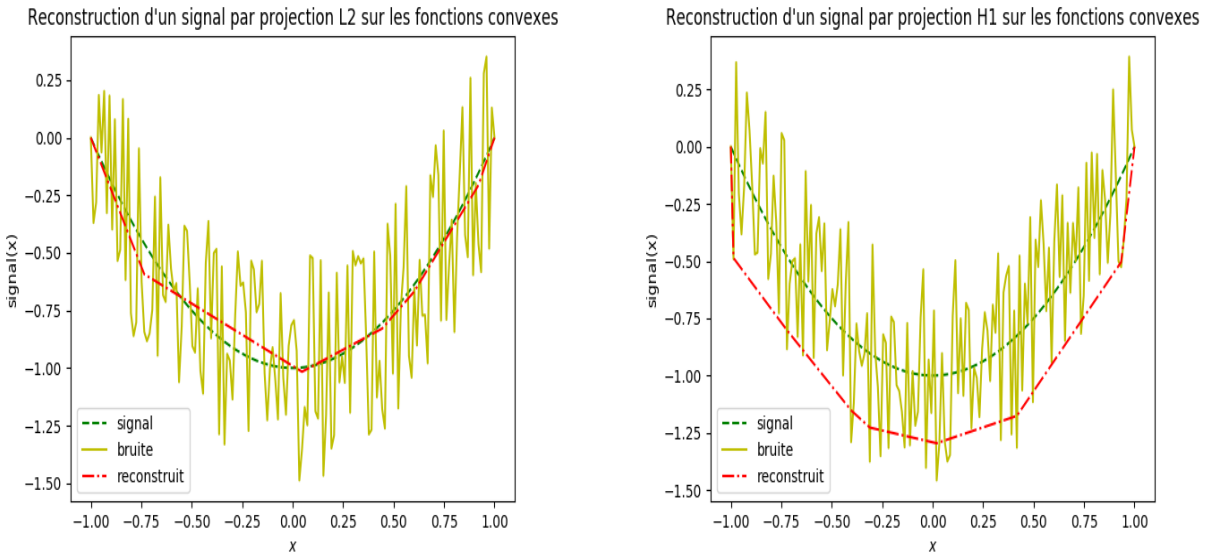

FIGURE 1 – Signal bruité et sa reconstruction

Note 2 (Attention : Cette partie est à rendre)**.**

L'enseignement n'ayant pas un examen théorique cette année, cette partie introduite par le chargé de cours permet de juger de la compréhension des techniques vues en cours.

Partie - 1- 1 *Problème continu*

On considère l'espace

$$H = H^1(]0,1[), \quad \big(\text{on admettra le plongement} \quad H^1(]0,1[) \hookrightarrow \mathcal{C}^0([0,1])\big)$$

muni de la norme $\|u\|_H = \int_0^1 (u(x)^2 + u'(x)^2)dx$, et on admet que cet espace est un espace de Hilbert. Une fonction $u \in \mathcal{C}^0([0,1])$ est convexe si

$$\forall x, y \in [0,1], \forall t \in [0,1], \quad u((1-t)x + ty) \leq (1-t)u(x) + tu(y).$$

Pour $u_0 \in H$ donné, on considère le problème consistant à trouver la projection de u_0 sur l'ensemble des fonctions convexes, où autrement dit la fonction convexe la plus proche de u_0 au sens de la norme sur H :

$$E = \inf_{u \in K} \|u - u_0\|_H \text{ où } K = \{u \in H \mid u \text{ est convexe}\}. \tag{1}$$

Q-1 : Pour $x, y, t \in [0,1]$ fixés, on note $K^{x,y,t} = \{u \in H \mid u((1-t)x + ty) \leq (1-t)u(x) + tu(y)\}$. Démontrer que $K^{x,y,t}$ est convexe et fermé dans H. En déduire que $K = \cap_{x,y,t \in [0,1]} K^{x,y,t}$ est également convexe fermé.

Indication : On admettra que si u_N converge vers u au sens de H (i.e. $\lim_{N\to\infty} \|u_N - u\|_H = 0$) alors u_N converge vers u uniformément.

Q-2 : Démontrer que l'infimum dans (1) est atteint en un unique $u^* \in H$ (penser au théorème de projection).

Partie - 1- 2 *Problème discret et étude de convergence*

Étant donné un niveau de discrétisation $N > 1$, on introduit $x_i = hi$ pour $0 \leq i \leq N+1$, où $h = 1/(N+1)$ et l'on pose

$$H_N = \{u \in H \mid \forall 0 \leq i \leq N, u|_{[x_i, x_{i+1}]} \text{ est affine}\}.$$
$$E_N = \inf_{u \in K_N} \|u - u_0\|_H \text{ où } K_N = \{u \in H_N \mid u \text{ est convexe}\}. \tag{2}$$

On considère la fonctionnelle $I_N : H \to H_N$ définie par

$$(I_N u)(x) = \frac{x - x_i}{h} u(x_i) + \frac{x_{i+1} - x_i}{h} u(x_{i+1}) \text{ si } x \in [x_i, x_{i+1}[.$$

En particulier, on a $(I_N u)(x_i) = u(x_i)$. On admet que cette fonctionnelle vérifie $\forall u \in H, \quad \lim_{N\to\infty} \|u - I_N u\|_H = 0$.

Q-3 : Montrer que $E_N \geq E$, et que l'infimum dans E_N est atteint en un unique $u_N^* \in H_N$.

Q-4 : **Le but de cette question est de démontrer que si u est convexe ($u \in K$) alors $I_N u$ l'est aussi ($I_N u \in K_N$).**

Q-4-1 : [*] Soit $u \in H_N$. Démontrer que u est convexe si et seulement si

$$\forall 1 \leq i \leq N, \quad u(x_i) - u(x_{i-1}) \leq u(x_{i+1}) - u(x_i) \tag{3}$$

Indication : on pourra utiliser le fait que u est convexe si et seulement si $u^+(x) := \lim_{t\to 0^+} \frac{u(x+t)-u(x)}{t}$ est bien définie (i.e. u est dérivable à droite) et si u^+ est croissante.

Q-4-2 : Montrer que (3) est équivalent à

$$\forall 1 \leq i \leq N, \quad u(x_i) \leq \frac{1}{2}\Big(u(x_{i+1}) + u(x_{i-1})\Big) \tag{4}$$

En déduire que si u est convexe, alors $I_N u$ l'est aussi.

Q-5 : Montrer en utilisant les questions précédentes
que $I_N u^* \in K_N$, que $E_N \leq \|I_N u^* - u_0\|$, puis que $\lim_{N\to\infty} E_N = E$.

Q-6 : Conclure que u_N^* converge faiblement vers u^*.

Partie - 2 *Mise en oeuvre et validations*

Note 3.

On souhaite construire explicitement le problème (2) et le résoudre numériquement. Pour cela on emploiera deux méthodes de résolution :

— la méthode d'Uzawa à traiter en séance,

— la méthode de pénalisation **à rendre.**

Sans nuire à la généralité on désignera par u^b la fonction à projeter qu'on supposera choisie telle que

$$u^b(0) = 0, \quad u^b(1) = 0.$$

Partie - 2- 1 *Construction du problème discret*

Q-7 : Soit $v \in K_N$ montrer que

$$\begin{aligned} \int_0^1 v(x)^2\, dx &= \sum_{i=0}^{N} \frac{h}{3}\left(v_i^2 + v_i v_{i+1} + v_{i+1}^2\right) \\ \int_0^1 v(x)'^2\, dx &= \sum_{i=0}^{N} \frac{1}{h}\left(v_i^2 - 2v_i v_{i+1} + v_{i+1}^2\right) \end{aligned} \tag{5}$$

Q-8 : Montrer que le problème (2) peut se mettre sous la forme

$$\begin{cases} E_N = \min\limits_{\mathbf{u}_N \in K_N} J_N(\mathbf{u}_N) \\ K_N = \left\{\mathbf{v} \in R^N : \varphi_i(\mathbf{v}) \leq 0 : \forall\ 1 \leq i \leq N\right\} \end{cases} \tag{6}$$

où J_N est à expliciter, et $\varphi_i(\mathbf{v}) = (A\ \mathbf{v})_i$ (ie i-ème composante du produit matrice-vecteur $A\ \mathbf{v}$) $\forall\ 1 \leq i \leq N$ avec A une matrice tridiagonale à préciser.

Q-8-1 : Calculer le gradient ∇J_N de J_N.

Q-8-2 : Donner l'expression du gradient des contraintes (ie $\nabla\varphi(\mathbf{v})$).

Q-9 : Implémentation

Q-9-1 : Implémenter les fonctionnelles J_N respectivement ∇J_N, à travers les deux fonctions PYTHON suivantes :

```
def J(U, x, Ub)
```

```
def DJ(U, x, Ub)
```

où U_b est l'interpolé P1-Lagrange de la fonction à projeter u^b : c'est-à-dire $U_b(i) = u^b(x_i),\ i = 0, \ldots, N+1$.
Et x contient les noeuds du maillage de $[0, 1]$, comme dans les Tps précédents.

Q-9-2 : Implémenter les contraintes et leur gradient à travers deux fonctions :

```
def Contraintes(U, x, Ub)
```

```
def GradContraintes(U, x, Ub)
```

Partie - 2- 2 *Résolution directe du problème et test de convergence*

Dans cette partie on fournit la fonction suivante à modifier si nécessaire :

Code Listing 1 – Fichier problemeModele.py

```
def problemeModele(signal, N, amplitude = 0.5):
    """
    Genere l'interpole P1, sur un maillage uniforme de [0,1] forme de N noeuds internes,
    d'un signal bruite issu d'un signal d'entree
    ENREE :
       signal    : une fonction definie sur [0,1] et a valeur reelle
       N         : nombre de noeuds internes du maillage
       amplitude : amplitude du bruit a introduire
    SORTIE:
       ub        : un signal bruite du signal d'entree sur un maillage de [0,1]
       x         :les noeuds du maillage considere de [0,1]
    """
    x = np.linspace(0,1, N+2)
    bruit =  (-1.0 + 2 * random.rand(N+2)) / 2
    ub = signal(x) + amplitude * bruit
    ub[0]  = 0
    ub[-1] = 0
    return ub, x
```

Q-10 : [**Validation 1/3**].

Ici on utilisera le module **`scipy.optimize.minimize`** de PYTHON (voir TP1 ou voir Listing 2).

Q-10-1 : Déterminer la solution du problème (6), pour un signal de votre votre choix, (*on prendra N = 60*).

Q-10-2 : Représenter sur la même figure le signal de départ, le signal bruité et le signal reconstruit en résolvant le problème (6).

Q-10-3 : Représenter sur une autre figure le vecteur des contraintes au point optimal obtenu. Identifier les contraintes qui ont été activées et commenter les résultats.

Q-11 : [**Validation 2/3 : qualité de l'approximation**]

En prenant un signal déjà convexe (par exemple $signal(x) = -\sin(\pi x)$ sans bruit (i.e amplitude = 0 dans la fonction **`problemeModele`** du Listing 1)), reprendre la question précédente et commenter le résultat.

Q-12 : [**Validation 3/3 : convergence de l'approximation**]

Sur un exemple au choix, faire varier N et représenter sur le même graphique les différentes solutions et commenter les résultats.

Partie - 2- 3 *Résolution par la méthode d'Uzawa*

On souhaite résoudre le problème (6) en utilisant les outils développés dans les Tps précédents.

Q-13 : [**Motivation pour la méthode d'Uzawa.**].

A quelle difficulté majeure serait-on confronté dans une mise en oeuvre de l'algorithme du gradient projeté à la résolution du problème (6) ?

Q-14 : [**Algorithme**]

A fin de palier à cette difficulté d'exploiter la méthode de projection sur le problème primal, on peut se pencher sur le problème dual. Il en découle alors entre autres méthodes, la méthode d'Uzawa.

Pour $\lambda = (\lambda_1, \ldots, \lambda_d)$, d représentatnt le nombre de contraintes d'inégalité (ici $d = N$), on note

$$L_\lambda(\mathbf{u}) = J_N(\mathbf{u}) + \sum_{i=1}^{d} \lambda_i \varphi_i(\mathbf{u}). \tag{7}$$

La méthode d'Uzawa consiste à remplacer le problème (6) par une suite de problèmes de minimisation sans contrainte portant sur les L_λ.

Pour un point de départ $\mathbf{u}^0$, un choix initial de $\lambda^0 = (\lambda_1^0, \ldots, \lambda_d^0)$, une tolérance ε, et un pas ρ donné, cet algorithme s'écrit de la façon suivante :

Méthode d'Uzawa
Initialiser le résidu r^0 à 1 et le compteur k à 0.
Tant que le résidu est plus grand que ε et que le compteur n'est pas trop grand :
- calculer $\mathbf{u}^{k+1}$ le minimiseur de $\mathbf{v} \mapsto L_{\lambda^k}(\mathbf{v})$ (voir Note 6),
- poser $\lambda^{k+1} = (\max(\lambda_i^k + \rho\varphi_i(\mathbf{u}^{k+1}), 0))_{i=1\ldots d}$,
- calculer le résidu $r^{k+1} = ||\mathbf{u}^{k+1} - \mathbf{u}^k||$,
- incrémenter le compteur.

Q-14-1 **:** Montrer que c'est un algorithme de gradient projeté appliqué au problème dual du problème (6).

Q-14-2 **:** Tel que présenté, cet algorithme assure-t-il la convergence de la suite des multiplicateurs de Lagrange ?

On admet dans toute la suite que le Lagrangien du problème (6) admet un point selle sur $\mathbb{R}^N \times \mathbb{R}_+^N$.

Q-14-3 **:** Donner en fonction de J_N et de A (matrice définissant les contraintes d'inégalités φ_i), une plage de choix du pas ρ assurant la convergence de l'algorithme d'Uzawa. (*On pourra aussi remarquer que J_N est quadratique et par conséquence associée à une matrice Q, et donner alors la plage de ρ en fonction de Q et A*).

Q-15 **:** [**Implementation et validation**]

Q-15-1 **:** On souhaite utiliser la méthode du gradient à pas fixe pour calculer à chaque étape le minimiseur de $\mathbf{v} \mapsto L_{\lambda^k}(\mathbf{v})$. Écrire une fonction **`gradient_fixe_lambda.py`** qui prend en arguments, entre autres, λ, ρ, ε et un point de départ $\mathbf{u}_0$, et qui retourne le minimiseur de $\mathbf{v} \mapsto L_\lambda(\mathbf{v})$.

Q-15-2 **:** Implémenter l'algorithme d'Uzawa à l'aide d'une fonction **`Uzawa.py`** dont on précisera les arguments. (*On s'inspirera des choix des arguments dans les algorithmes développés dans les Tps précédents*).

Q-15-3 **:** Créer un script **`script_uzawa.py`** et résoudre le problème de minimisation sous contrainte $\min_{\mathbf{u} \in K_N} J_N(\mathbf{u})$ à l'aide de l'algorithme d'Uzawa. On prendra $\mathbf{u}^0$ et λ^0 nuls. On prendra ρ dans un intervalle $[0, \rho_{\max}]$ où $\rho_{\max}$ sera à déterminer.

Q-16 **:** [**Analyse**]

Q-16-1 **:** Que se passe-t-il si l'on augmente le pas ρ ?

Q-16-2 **:** Peut-on modifier l'algorithme d'Uzawa de sorte à faire varier ρ ? Y' a-t-il intérêt à le faire ?

Q-16-3 **:** Dans le cas particulier du problème (6) peut-on déterminer une bonne valeur fixe du pas ρ ?

Partie - 2- 4 *Résolution par la méthode de pénalisation (***partie à rendre***)*

Note 4 (Attention)**.**

Cette partie est à rendre.

Note 5.

- La fin de l'exercice sur la méthode de pénalisation du TP2 a mis en évidence des comportements différents suivant les valeurs du paramètre de pénalisation.
- Il est donc judicieux de mettre en place un algorithme itératif permettant d'ajuster le paramètre de pénalisation à chaque itération.
- On se propose ici de mettre en oeuvre un tel algorithme et le comparer avec la méthode d'Uzawa sous la main. Mais pour que cette comparaison soit être possible, l'algorithme dans son implémentation devra retourner à chaque itération une approximation du multiplicateur de Lagrange.

On procède comme dans la méthode d'Uzawa. Pour $\eta \in \mathbb{R}^*$, on note

$$L_\eta(\mathbf{u}) = J_N(\mathbf{u}) + \frac{1}{\eta}\sum_{i=1}^{d}\Big(\max(\varphi_i(\mathbf{u}),0)\Big)^2.$$

La méthode de pénalisation avec ajustement du paramètre de pénalisation s'écrit alors :

Méthode de pénalisation
Initialiser le résidu r^0 à 1, le compteur k à 0 et η_0 à 1.
Tant que le résidu est plus grand que ε et que le compteur n'est pas trop grand :
- calculer $\mathbf{u}^{k+1}$ le minimiseur (*approché*) de $\mathbf{v} \mapsto L_{\eta_k}(\mathbf{v})$: voir Note 6
- mettre à jour le multiplicateur de Lagrange λ^{k+1} : voir Note 7,
- choisir un nouveau paramètre de pénalisation $\eta_{k+1} < \eta_k$: voir Note 7,
- calculer le résidu $r^{k+1} = ||\mathbf{u}^{k+1} - \mathbf{u}^k||$,
- incrémenter le compteur.

Q-17 **:** Modifier la fonction **`penalisation.py`** du TP2 afin de mettre en oeuvre cet algorithme.

Q-18 **:** Créer un script **`script_penalisation.py`** et résoudre le problème (6)

Q-19 **:** **Importance de l'ajustement de paramètre de pénalisation**

Mettre en place une expérience permettant de justifier cette nécessité d'ajuster le paramètre de pénalisation dans l'algorithme proposé.
*On pourra comparer la solution et l'effort de son calcul par la méthode de pénalisation à ceux obtenus en résolvant directement le problème pénalisé pour une certaine valeur du paramètre de pénalisation à préciser. (***La pertinence du choix de cette valeur sera appréciée***).*

Q-20 **:** **Comparaison avec la méthode d'Uzawa** .

Comparer cet algorithme de pénalisation avec la méthode d'Uzawa pour le problème considéré. *On insistera*
- *sur la qualité de la solution,*
- *la difficulté à obtenir cette solution qu'on pourra mesurer par le nombre d'itérations effectuées et le temps mis pour la résolution.*
- *les valeurs approchées des multiplicateurs de Lagrange obtenues.*

Q-21 : Vers une meilleure association avec la méthode d'Uzawa.

Q-21-1 : Proposer un algorithme d'Uzawa dans lequel la méthode de pénalisation servirait d'initialisation.

Q-21-2 : Justifier le bien fondé d'un tel algorithme.

Q-21-3 : Valider cet algorithme sur le problème (6).

Partie - 3 *Queques notes et utilitaires*

Note 6 (Résolution du problème sans contrainte)**.**

La résolution du problème de minimisation sans contrainte peut se faire soit à l'aide des fonctions **`scipy.optimize.minimize ...`** de **`Python`**, soit en faisant recours à l'une des méthodes vues lors des Tps précédents. La donnée initiale sera fournie par l'itéré précédent.

Note 7 (Mise à jour des paramètres : η_{k+1} et $\lambda^{k+1} = (\lambda_1^{k+1}, \ldots, \lambda_d^{k+1})$)**.**

1. **Paramètre de pénalisation** :
   La mise à jour du nouveau paramètre de pénalisation peut se faire de la manière suivante :
$$\eta_{k+1} = \frac{\eta_k}{100}. \tag{8}$$
2. **Multiplicateur de Lagrange dans la méthode pénalisation** :
   La méthode de pénalisation ne dépend pas du multiplicateur de Lagrange. On peut néanmoins estimer le multiplicateur de Lagrange par la formule suivante :
$$\lambda_i^{k+1} = \frac{2}{\eta_k} \max(\varphi_i(\mathbf{u}^k), 0) \quad i = 1, \ldots, d. \tag{9}$$

Code Listing 2 – Exemple de résolution directe

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import numpy.random as random

N = 60

signal = lambda x: - np.sin(np.pi * x)

ub, x = problemeModele(signal, N, 0.1)

Jf = lambda u : (J(u, x, ub[1:-1])
DJf = lambda u : (DJ(u, x, ub[1:-1])
cons = ({'type': 'ineq',
         'fun' : lambda u: -Contraintes(u, x, ub[1:-1]),
         'jac' : lambda u: -GradContraintes(u, x, ub[1:-1])})

u = np.zeros(N)  # un peut aussi prendre u = ub[1:-1])

res = minimize(Jf, u, method='SLSQP', jac=DJf, constraints=cons, tol=1e-12,
               options={'disp': True,'maxiter':5000})


# res.x contient la solution; On peut la representer graphiquement ....
...
...
```