UNLocBoX : Toolbox MATLAB d'Optimisation Convexe

Méthodes de Proximal Splitting

DONGMO TCHOUMENE ANITA BELVIANE - **22W2184**
DONFACK SYNTHIA CALORINE - **22U2073**
BOKOU-BOUNA-ANGE-LARISSA - **22W2188**
JIATSA ROMMEL JUNIOR - **22T2906**

Université de Yaounde I - Master I Data Science

22 octobre 2025

Plan de la présentation

Qu'est-ce que UNLocBoX ?

- **UNLocBoX** : boîte à outils MATLAB pour l'optimisation convexe
- Basée sur les **méthodes de proximal splitting**
- Développée à l'**EPFL** (Lausanne, Suisse)
- Open source et bien documentée
- Efficace pour les problèmes de grande dimension

Problème résolu par UNLocBoX

Forme générale du problème :

$$\min_{x \in \mathbb{R}^N} \sum_{n=1}^{K} f_n(x)$$

Caractéristiques :

- Chaque f_n est une fonction convexe
- Les fonctions peuvent être différentiables ou non
- Somme de termes "simples" (régularisation, fidélité aux données, contraintes)

Pourquoi l'optimisation convexe ?

- De nombreux problèmes en **Machine Learning** et **traitement du signal**
- Reconstruction d'images, débruitage, régression sparse
- Garantie de convergence vers un minimum global
- Algorithmes classiques (Newton, gradient) : trop coûteux pour le big data

Limitations des méthodes classiques

- **Gradient descent** : nécessite différentiabilité
- **Méthodes du second ordre** : complexité $\mathcal{O}(N^3)$
- Inadaptées aux fonctions non-différentiables (norme ℓ_1, TV)
- Proximal splitting : **scalable** et **flexible**

Opérateur proximal : définition

Définition :

$$\mathrm{prox}_f(x) = \arg\min_y \left(\frac{1}{2}\|x - y\|^2 + f(y) \right)$$

Propriétés :

- Généralise la projection sur un ensemble convexe
- Solution unique pour toute fonction convexe semi-continue
- Permet de traiter les fonctions non différentiables

Proximal splitting : principe

Idée centrale :

- Découpler la fonction objective en termes simples
- Résoudre chaque terme séparément via son opérateur proximal
- Alterner entre les différents opérateurs
- Convergence garantie vers la solution optimale

Avantages :

- Complexité $\mathcal{O}(N)$ par itération
- Parallélisation possible

Structure générale

Composantes principales :

1. **Solvers** : implémentent les algorithmes d'optimisation
2. **Opérateurs proximaux** : fonctions prédéfinies
3. **Fichiers de démonstration** : exemples d'utilisation
4. **Fonctions utilitaires** : outils auxiliaires

Solvers disponibles

Solvers spécifiques (2 fonctions) :

- `forward_backward` (FISTA)
- `douglas_rachford`
- `admm` / `sdmm`
- `chambolle_pock`

Solvers généraux (K fonctions) :

- `generalized_forward_backward`
- `ppxa`

Fonction automatique : `solvep`

Opérateurs proximaux prédéfinis

- `prox_l1` : norme ℓ_1 (soft-thresholding)
- `prox_l2` : norme ℓ_2
- `prox_tv` : variation totale (TV)
- `prox_nuclear` : norme nucléaire
- `proj_b2` : projection sur boule ℓ_2
- `proj_linear_eq` : projection sur contraintes linéaires

Définition des fonctions

Fonction différentiable :

```
f.eval = @(x) norm(A*x - y)^2;
f.grad = @(x) 2*A'*(A*x - y);
f.beta = 2*norm(A)^2;  % Lipschitz
```

Fonction non différentiable :

```
f.eval = @(x) lambda*norm(x,1);
f.prox = @(x,T) prox_l1(x, lambda*T);
```

Résolution d'un problème

Exemple : Régression Lasso

$$\min_x \|Ax - y\|_2^2 + \lambda \|x\|_1$$

```
% Fonction 1 : terme quadratique
f1.grad = @(x) 2*A'*(A*x - y);
f1.beta = 2*norm(A)^2;

% Fonction 2 : norme L1
f2.prox = @(x,T) prox_l1(x, lambda*T);

% R solution
sol = forward_backward(y, f1, f2, param);
```

Paramètres des solvers

Paramètres courants :

- `param.maxit` : nombre max d'itérations (défaut : 200)
- `param.tol` : tolérance de convergence (défaut : 10^{-2})
- `param.gamma` : pas de temps (calculé automatiquement)
- `param.verbose` : niveau de log (0, 1 ou 2)

Application 1 : Inpainting d'image

Problème : Reconstruction d'une image avec pixels manquants
Formulation :

$$\min_x \|x\|_{TV} \quad \text{s.c.} \quad \|Ax - y\|_2 \leq \epsilon$$

- A : opérateur de masque
- y : pixels observés
- Régularisation par variation totale (contours nets)

Solver : Douglas-Rachford

Application 2 : Débruitage par régularisation

Problème : Débruitage de signal/image
Formulation :

$$\min_x \|x - y\|_2^2 + \lambda \|x\|_1$$

- y : signal bruité
- Régularisation ℓ_1 : favorise la parcimonie
- Trade-off via λ

Solver : Forward-Backward (FISTA)

Application 3 : Compressed Sensing

Problème : Reconstruction de signal sparse sous-échantillonné
Formulation :

$$\min_x \|x\|_1 \quad \text{s.c.} \quad Ax = y$$

- A : matrice de mesure (sous-déterminée)
- Contrainte d'égalité stricte
- Promotion de la parcimonie

Solver : Douglas-Rachford avec projection

Avantages de UNLocBoX

- **Flexibilité** : gestion de multiples fonctions et contraintes
- **Efficacité** : complexité linéaire, adapté au big data
- **Modularité** : ajout facile de nouveaux opérateurs/solvers
- **Documentation complète** : guide utilisateur + exemples
- **Open source** : gratuit et personnalisable

Comparaison avec d'autres outils

Outil	**Flexibilité**	**Scalabilité**	**Contrôle**
CVX	Haute	Faible	Faible
UNLocBoX	Haute	Haute	Élevé
Gradient classique	Faible	Moyenne	Élevé

UNLocBoX : compromis optimal entre facilité et performance

Conclusion et perspectives

UNLocBoX est une boîte à outils puissante et flexible permettant de résoudre efficacement de nombreux problèmes convexes. S'appuyant sur les méthodes modernes de splitting et proximalité, elle est documentée et adaptée à la recherche et l'enseignement.

Points clés :

- UNLocBoX = outil moderne pour l'optimisation convexe
- Méthodes proximales : scalables et efficaces
- Applications variées en Data Science
- Extensible et personnalisable