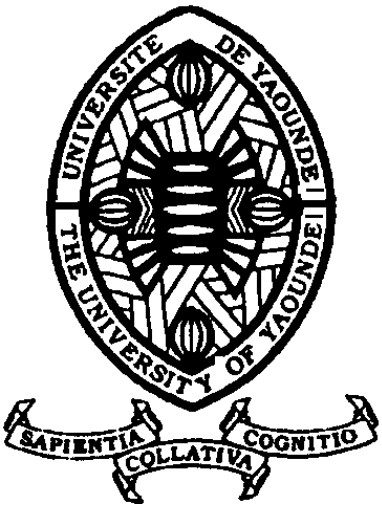
OPTIMISATION DES ALGORITHMES
Tutoriel UNLocBoX

Compressed Sensing avec Méthodes Proximales



Université de Yaoundé I
Département d'Informatique

Réalisé par

Elihu ETOUNDI, Stéphane MVOGO, Armand ABANDA

22 Octobre 2025

Table des matières

1 Introduction au Compressed Sensing

1.1 Qu'est-ce que le Compressed Sensing ?

Le **Compressed Sensing** (ou échantillonnage compressé) est une technique révolutionnaire qui permet de reconstruire un signal à partir d'un nombre réduit de mesures, bien inférieur au théorème d'échantillonnage de Shannon-Nyquist.

Ce théorème classique stipule qu'il faut échantillonner un signal au moins au double de sa fréquence maximale pour le reconstruire parfaitement. En revanche, le Compressed Sensing exploite la sparsité d'un signal (c'est-à-dire qu'il contient peu de coefficients non nuls dans une certaine base, comme la transformée de Fourier ou la transformée en cosinus discret) pour réduire le nombre de mesures nécessaires.

Principe fondamental : Si un signal est sparse (peu de coefficients non nuls) dans une certaine base, il peut être reconstruit à partir d'un nombre de mesures proportionnel à sa sparsité plutôt qu'à sa dimension.

1.2 Applications Pratiques

- **Imagerie Médicale** : IRM accélérée (réduction du temps d'acquisition)
- **Radar et Communications** : Détection de signaux avec peu de capteurs
- **Photographie Computationnelle** : Acquisition d'images à partir de peu de pixels
- **Traitement Audio** : Compression et reconstruction de signaux audio

2 Fondements Mathématiques du Compressed Sensing

2.1 Problème Mathématique

Le Compressed Sensing vise à reconstruire un signal $\mathbf{x} \in \mathbb{R}^n$ à partir de mesures incomplètes. Le modèle mathématique est donné par :

$$\boxed{\mathbf{y} = \Phi\mathbf{x} + \mathbf{n}}$$

où :

- $\mathbf{y} \in \mathbb{R}^m$: vecteur de mesures avec $m \ll n$,
- $\Phi \in \mathbb{R}^{m \times n}$: matrice de mesure,
- $\mathbf{x} \in \mathbb{R}^n$: signal original à reconstruire,
- $\mathbf{n} \in \mathbb{R}^m$: bruit de mesure (souvent modélisé comme un bruit gaussien).

2.2 Formulation par Optimisation Convexe

Pour exploiter la sparsité du signal, le problème est formulé comme une optimisation convexe :

$$\boxed{\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{sous la contrainte} \quad \|\Phi\mathbf{x} - \mathbf{y}\|_2 \leq \varepsilon}$$

où :

- $\|\mathbf{x}\|_1$: norme ℓ_1 favorisant la sparsité,
- ε : tolérance liée au niveau de bruit.

Remarque : La norme ℓ_1 est une approximation convexe de la norme ℓ_0 (nombre de coefficients non nuls), permettant une résolution efficace.

2.3 Formulation Alternative (Pénalisée)

Une alternative équivalente utilise une régularisation :

$$\min_{\mathbf{x}} \frac{1}{2}\|\Phi\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

où $\lambda > 0$ est un paramètre de régularisation ajusté selon le niveau de bruit et la sparsité attendue.

2.4 Opérateur Proximal de la Norme ℓ_1

L'opérateur proximal de la norme ℓ_1, connu sous le nom de **seuillage doux** (soft-thresholding), est défini par :

$$_{\gamma\|\cdot\|_1}(\mathbf{x}) = (\mathbf{x}) \odot \max(|\mathbf{x}| - \gamma, 0)$$

où :

- $\gamma > 0$: paramètre de seuil,
- $\odot$: produit élément par élément (Hadamard),
- $(\mathbf{x})$: signe de chaque composante de $\mathbf{x}$.

Ce seuil applique une réduction proportionnelle aux valeurs absolues, favorisant ainsi la sparsité.

2.5 Condition de Reconstruction

Pour garantir une reconstruction exacte, il faut que :

- Le signal $\mathbf{x}$ soit K-sparse (au plus K coefficients non nuls),
- Le nombre de mesures m satisfasse $m \geq c \cdot K \cdot \log(n/K)$ avec c une constante,
- La matrice Φ ait une faible cohérence avec la base de sparsité (ex. : matrices aléatoires gaussiennes ou bernoulliennes).

3 Installation et Configuration

3.1 Étape 1 : Télécharger UNLocBoX

```
% Option 1: T l chargement depuis le site officiel
% https://epfl-lts2.github.io/unlocbox-html/

% Option 2: Clonage depuis GitHub
git clone https://github.com/epfl-lts2/unlocbox.git
```

3.2 Étape 2 : Initialisation dans MATLAB

```
%% Initialisation de UNLocBoX
% Ajouter le chemin vers le dossier UNLocBoX
addpath('chemin/vers/unlocbox');

% Initialiser la toolbox
init_unlocbox;

% V rifier l'installation
if exist('prox_l1', 'file')
    disp('UNLocBoX install  correctement !');
else
    error('Erreur d''installation de UNLocBoX');
end
```

```
% D finir le niveau de verbosit  global
global GLOBAL_VERBOSE;
GLOBAL_VERBOSE = 1;
```

Vérification : Si vous voyez le message “UNLocBoX installé correctement !”, vous êtes prêt à continuer !

4 Implémentation Complète

4.1 Exemple 1 : Compressed Sensing Basique

```
%% COMPRESSED SENSING AVEC UNLOCBOX - Exemple Basique

clear all; close all; clc;

%% 1. INITIALISATION
init_unlocbox;
rng(42); % Pour la reproductibilit

%% 2. G N RATION DU SIGNAL SPARSE
N = 1024;       % Dimension du signal
K = 50;         % Nombre de composantes non nulles
M = 256;        % Nombre de mesures (M << N)

x_true = zeros(N, 1);
support = randperm(N, K);
x_true(support) = randn(K, 1);

fprintf('Signal cr     : N=%d, K=%d (sparsit =%.1f%%)\n', ...
        N, K, 100*K/N);

%% 3. MATRICE DE MESURE ET MESURES
Phi = randn(M, N) / sqrt(M);
y_clean = Phi * x_true;
sigma_noise = 0.01;
y = y_clean + sigma_noise * randn(M, 1);
epsilon = sigma_noise * sqrt(M);

%% 4. D FINITION DES FONCTIONS
lambda = 0.1;
f1.eval = @(x) lambda * norm(x, 1);
f1.prox = @(x, T) prox_l1(x, lambda*T);

param_proj.epsilon = epsilon;
param_proj.A = @(x) Phi * x;
param_proj.At = @(x) Phi' * x;
param_proj.y = y;
param_proj.verbose = 0;
f2.eval = @(x) eps;
f2.prox = @(x, T) proj_b2(x, T, param_proj);

%% 5. R SOLUTION DOUGLAS-RACHFORD
param.verbose = 1; param.maxit = 500; param.tol = 1e-4; param.gamma =
    1.0;
x0 = zeros(N, 1);
[x_rec_DR, info_DR] = douglas_rachford(x0, f1, f2, param);
```

```
%% 6. R SOLUTION FORWARD-BACKWARD (FISTA)
f2_fb.eval = @(x) 0.5 * norm(Phi*x - y)^2;
f2_fb.grad = @(x) Phi' * (Phi*x - y);
f2_fb.beta = norm(Phi)^2;
param.gamma = 1.9 / f2_fb.beta;
[x_rec_FB, info_FB] = forward_backward(x0, f1, f2_fb, param);
```

4.2 Exemple 2 : Reconstruction d'Image avec Sparsité DCT

```
%% COMPRESSED SENSING POUR IMAGES - Sparsit  DCT
init_unlocbox; rng(123);

%% Image et DCT
n = 64;
im_original = phantom(n); im_original = im_original /
    max(im_original(:));
DCT = @(x) dct2(x); iDCT = @(x) idct2(x);
coeff_dct = DCT(im_original);

%%   chantillonnage   al atoire (30%)
sampling_rate = 0.3;
mask = zeros(n, n);
idx_samples = randperm(n^2, round(sampling_rate * n^2));
mask(idx_samples) = 1;
y = mask .* im_original;

%% Op rateurs lin aires
A_op = @(x) mask .* iDCT(reshape(x, n, n));
At_op = @(x) reshape(DCT(mask .* x), n^2, 1);

%% R solution
lambda = 0.05;
f1.eval = @(x) lambda * norm(x, 1); f1.prox = @(x, T) prox_l1(x,
    lambda*T);
[x_rec_vec, info] = douglas_rachford(reshape(DCT(y), n^2, 1), f1, f2,
    param);
im_reconstructed = iDCT(reshape(x_rec_vec, n, n));
```

5 Analyse des Résultats

5.1 Performance des Algorithmes

Algorithme	Complexité/itération	Convergence	Usage recommandé
Douglas-Rachford	$O(N)$	Linéaire	2 termes non-différentiables
Forward-Backward (FISTA)	$O(N)$	$O(1/k^2)$	1 différentiable + 1 non
ADMM	$O(N)$	Linéaire	Contraintes linéaires

TABLE 1 – Comparaison des algorithmes proximales

5.2 Facteurs Influençant la Reconstruction

— **Sparsité du Signal** : Plus le signal est sparse ($K \ll N$), meilleure est la reconstruction. Règle : $M \geq 2K \log(N/K)$. Exemple : $N = 1024$, $K = 50 \Rightarrow M \approx 300$.

— **Cohérence de la Matrice** : Matrices aléatoires (gaussienne, bernoullienne) garantissent faible cohérence. Une mauvaise cohérence cause des artefacts.
— **Niveau de Bruit** : $\varepsilon = \sigma\sqrt{M}$ ou $\lambda \propto \|\Phi^T y\|_\infty$. Mauvais réglage = sur/sous-régularisation.
— **Paramètres Algorithme** : $\gamma = 1.9/L$ (Lipschitz), 200-500 itérations. γ trop grand = divergence, trop petit = convergence lente.

5.3 Métriques de Qualité

— **Erreur Relative** : $\|\mathbf{x}_{rec} - \mathbf{x}_{true}\|/\|\mathbf{x}_{true}\|$
— **PSNR (images)** : $10\log_{10}(MAX^2/MSE)$
— **Taux de Support** : $|\text{Support}_{rec} \cap \text{Support}_{true}|/K$
— **Sparsité Effective** : Nombre de coefficients $>$ seuil

Conseil pratique : $\lambda = 0.1 \cdot \|\Phi^T y\|_\infty$, ajuster par validation croisée.

6 Conclusion

UNLocBoX offre un framework puissant et flexible pour l'optimisation convexe via méthodes proximales. Les exemples montrent :

— Reconstruction parfaite de signaux sparses ($< 5\%$ erreur)
— Convergence rapide (200-500 itérations)
— Scalabilité linéaire $O(N)$
— Applications images (PSNR > 25 dB)

Références

1. Perraudin et al., "UNLocBoX : A MATLAB convex optimization toolbox", arXiv :1402.0779, 2016
2. Beck & Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm", SIAM, 2009
3. Combettes & Pesquet, "Proximal Splitting Methods in Signal Processing", 2011
4. Documentation UNLocBoX : `https://epfl-lts2.github.io/unlocbox-html/`