

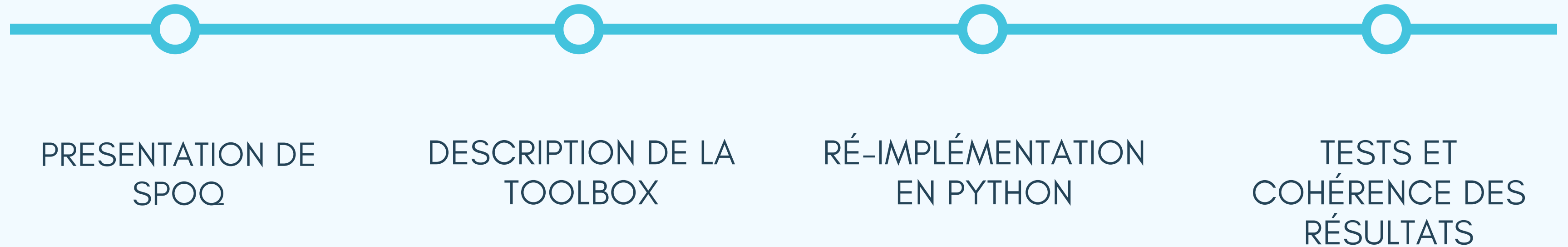
CentraleSupélec

SPARSE MODELS

**Ré-implémentation de la
toolbox Spog sur Python**

Anas LAAROUSSI
Charaf BOUCHTIOUI
Paul JOIN-LAMBERT
Yasmine BENNANI
Mamoun ALAOUI

PLAN DE LA PRESENTATION





1. PRÉSENTATION DE SPOQ

Problématique

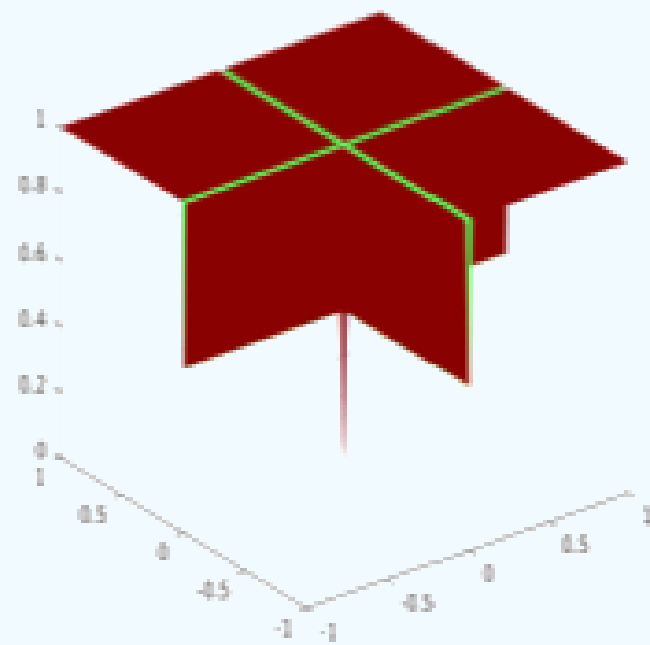
● Modèle d'observation

- Estimation d'un signal parcimonieux $x \in \mathbb{R}^N$ à partir d'observations $y \in \mathbb{R}^M$ dégradées, suivant le modèle:
 $y = Hx + b$ où $H \in \mathbb{R}^{M \times N}$ est une matrice de mesure et $b \in \mathbb{R}^M$ un bruit additif.

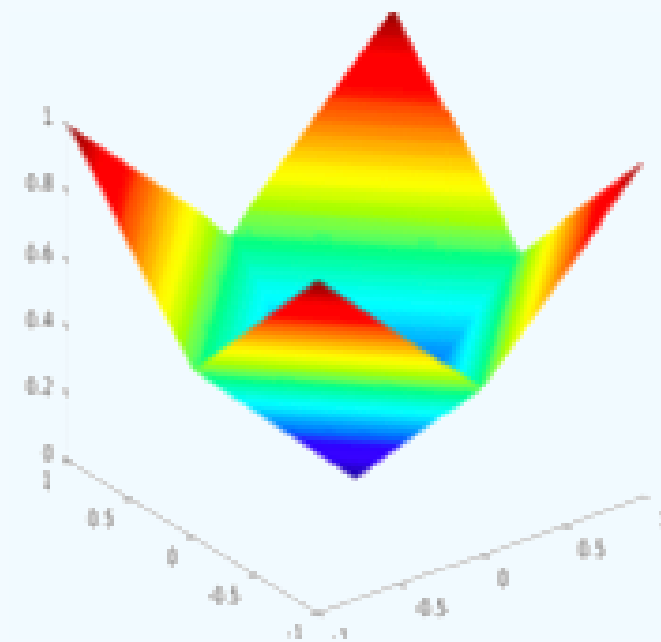
● Approche variationnelle

- Pour estimer \hat{x} , une stratégie efficace revient à résoudre:
minimiser $\Psi(x)$ tel que $S = \{x \in \mathbb{R}^N \mid \|Hx - y\| \leq \xi\}$ (1)
où : $\Psi : \mathbb{R}^N \rightarrow]-\infty, +\infty]$ est une fonction de régularisation qui renforce la parcimonie et $\xi > 0$ est un paramètre qui dépend du bruit

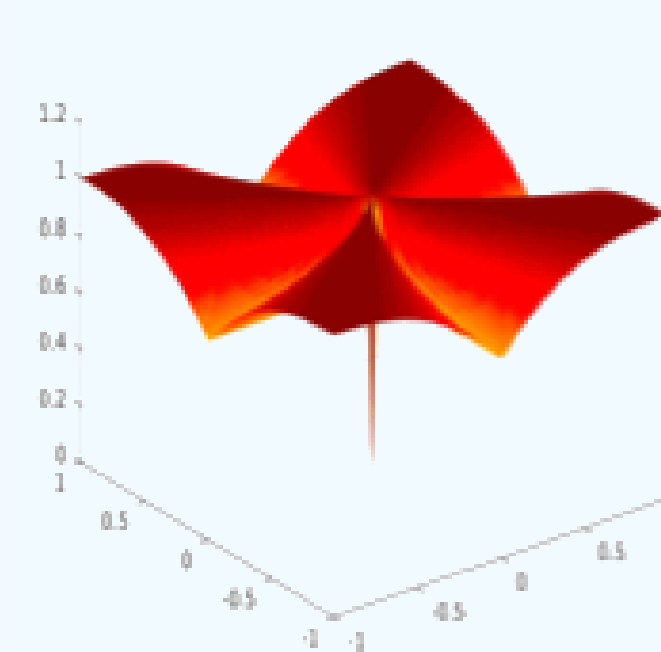
Difficulté: quel choix pour ψ ?



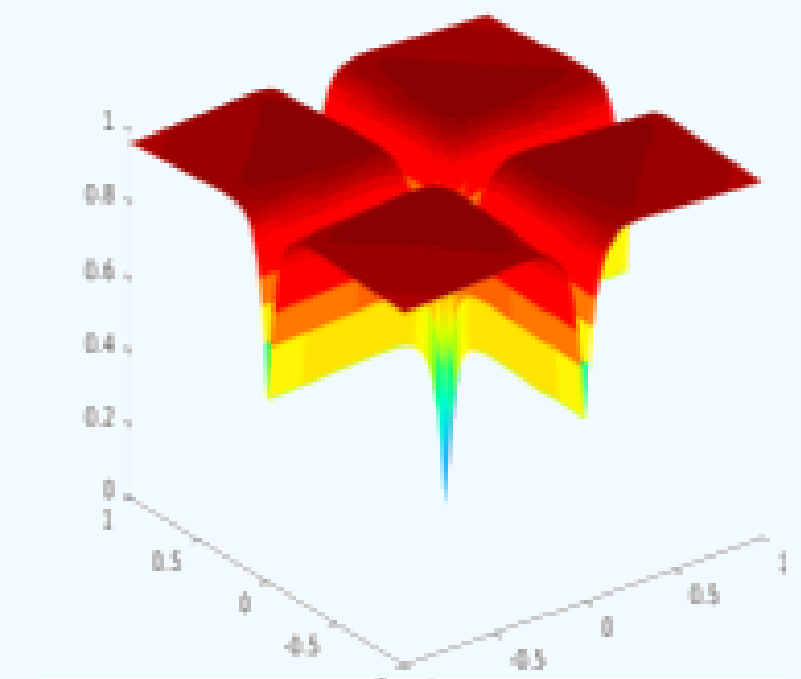
Indice de parcimonie l_0 :
pénalité de référence mais usage compliqué en pratique



Norme l_1 : pénalité standard permettant de se placer dans le cadre convexe mais pas la meilleure approximation de l_0



soot l_1/l_2 : rapport de normes pour retrouver la propriété d'invariance en échelle ($||\lambda x||_0 = ||x||_0$) de l_0



spoq l_p/l_q :
généralisation de soot à des quasi-norme l_p et l_q pour mieux approximer l_0

Approche proposée

Forme lissée de rapports de normes ℓ_p/ℓ_q

$$(\forall \mathbf{x} \in \mathbb{R}^N) \quad \Psi(\mathbf{x}) = \log \left(\frac{(\ell_{p,\alpha}^p(\mathbf{x}) + \beta^p)^{1/p}}{\ell_{q,\eta}(\mathbf{x})} \right)$$

Le problème d'optimisation (1) est non convexe mais Ψ proposée a un caractère gradient lipshitzien et est majorée localement par une forme quadratique --> proposition d'un algorithme Forward-Backward à métrique variable localement ajustée:

$\mathbf{x}_0 \in \mathbb{R}^N$, $B \in \mathbb{N}^*$, $\theta \in]0, 1[$, $(\gamma_k)_{k \in \mathbb{N}} \in]0, 2[$

Pour $k = 0, 1, \dots$:

 Pour $i = 1, \dots, B$:

- Construire $\rho_{k,i}$ selon (5).
- Construire $\mathbf{A}_{k,i} = \mathbf{A}_{q,\rho_{k,i}}(\mathbf{x}_k)$ selon (4).
- $\mathbf{z}_{k,i} = \mathbf{P}_{\mathbf{A}_{k,i}, \|\mathbf{H} \cdot - \mathbf{y}\| < \xi}(\mathbf{x}_k - \gamma_k (\mathbf{A}_{k,i})^{-1} \nabla \Psi(\mathbf{x}_k))$
- Si $\mathbf{z}_{k,i} \in \mathcal{B}_{q,\rho_{k,i}}$: Fin pour

$\mathbf{x}_{k+1} = \mathbf{z}_{k,i}$



2. DESCRIPTION DE LA TOOLBOX

Architecture globale

FB_PPXALpLq.m

Algorithme Forward-Backward à métrique variable

Initialisation

pds.m : Initilisation

Fcost.m : forme lissée de rapports de normes l_p/l_q (SPOQ) sur les x_{k_old} initialisé

ComputeLipschitz.m : Matrice $A_{q,p}$ de l'article

Boucle for

gradlplq.m : gradient de la fonction lissée l_p sur l_q

proxPPXApus.m : opérateur de proximité utilisant l'algorithme PPXA+

condlplq.m : métrique pour l'algorithme Forward-Backward à métrique variable.

proxl1.m
proxl2.m

Lpsmooth.m
Lqsmooth.m

proxB.m
proxl2.m
norm2.m

Lpsmooth.m



3. RÉ-IMPLÉMENTATION EN PYTHON



Structure du code

Implémentation Matlab

Toolbox:

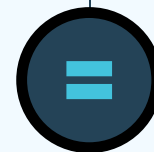
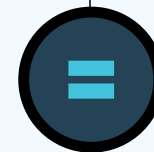
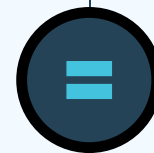
- | Tools
 - | proxPPXAplus.m
 - | proxl2.m
 - (...)
 - | ComputeLipshitz.m

Start simulations:

- | Run_SPOQ_Recovery.m

Run code:

- | Load_SPOQ_Data_User.m
- | Load SPOQ_Data_Simulated.m
- (...)



Implémentation Python

Toolbox:

- | utils.py
 - | proxPPXAplus()
 - | proxl2()
 - (...)
 - | ComputeLipshitz()

Start simulations:

- | spoq.py

Run code:

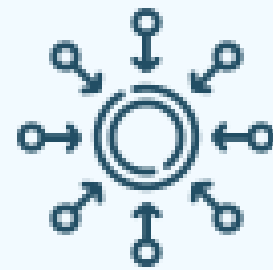
- | main.py
- | parser.py

Mode d'utilisation et flexibilité

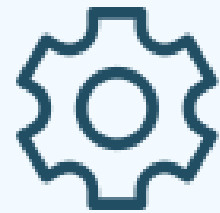
Besoin



Code fonctionnel et robuste



Architecture claire et fonctions centralisées et groupés



Ergonomie et choix des paramètres



Clarté du code et documentation



Implémentation proposée

Vérification et tests effectués
(cf section 3.)

Fonction `utils.py` unique regroupant les fonctions de la toolbox

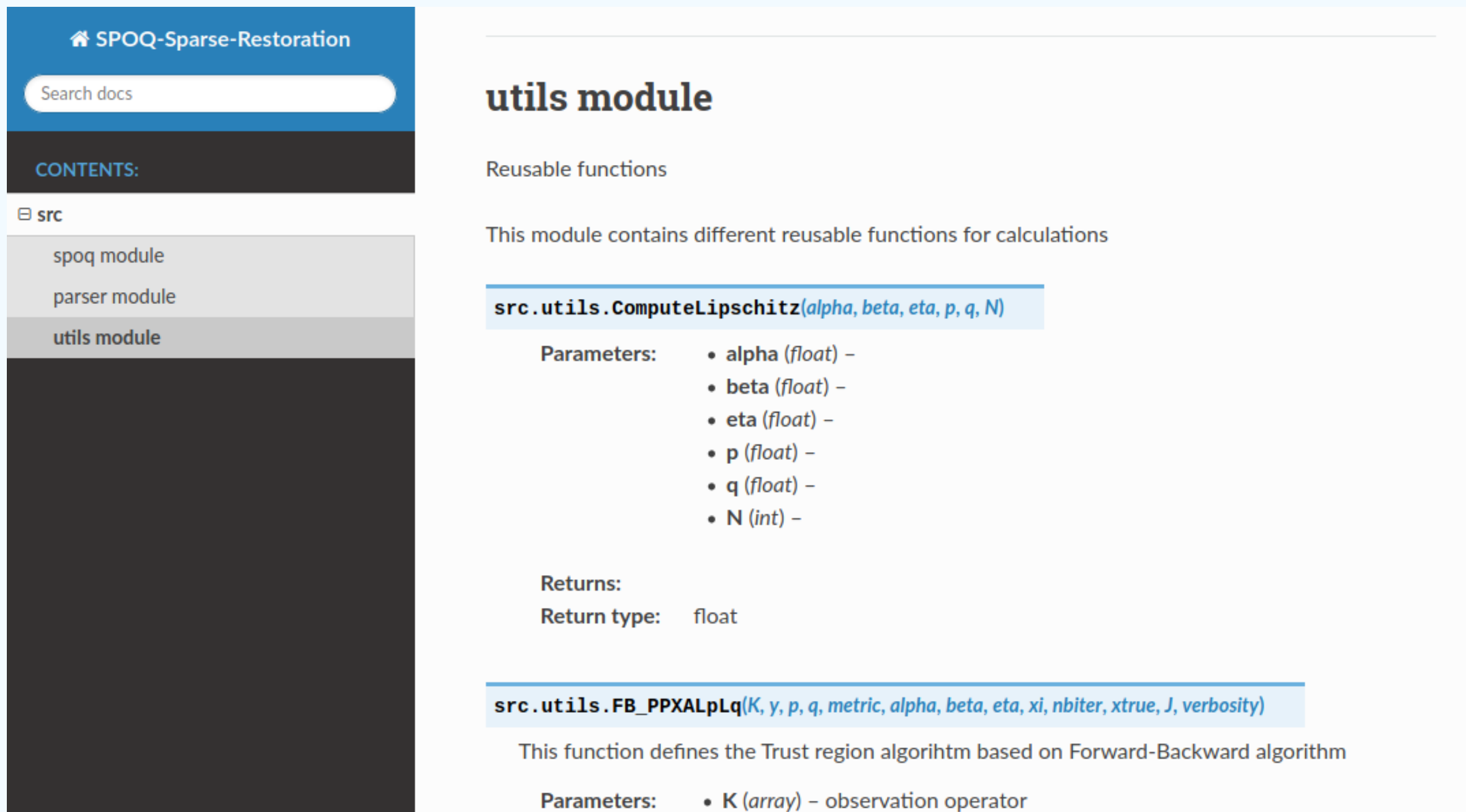
Parser permettant de choisir les paramètres au moment de l'appel au main

Code commenté et documentation disponible

Documentation

- Documentation du code aux normes Numpy
- Generation automatique de la documentation avec sphinx
- Deployment de la documentation sous forme de site statique sur gitlab pages

URL : <https://anas.laaroussi.pages-student.centralesupelec.fr/spoq/>



SPOQ-Sparse-Restoration

Search docs

CONTENTS:

- src
 - spoq module
 - parser module
 - utils module**

utils module

Reusable functions

This module contains different reusable functions for calculations

`src.utils.ComputeLipschitz(alpha, beta, eta, p, q, N)`

Parameters:

- `alpha (float)` –
- `beta (float)` –
- `eta (float)` –
- `p (float)` –
- `q (float)` –
- `N (int)` –

Returns:

Return type: float

`src.utils.FB_PPXALpLq(K, y, p, q, metric, alpha, beta, eta, xi, nbiter, xtrue, J, verbosity)`

This function defines the Trust region algorithm based on Forward-Backward algorithm

Parameters:

- `K (array)` – observation operator

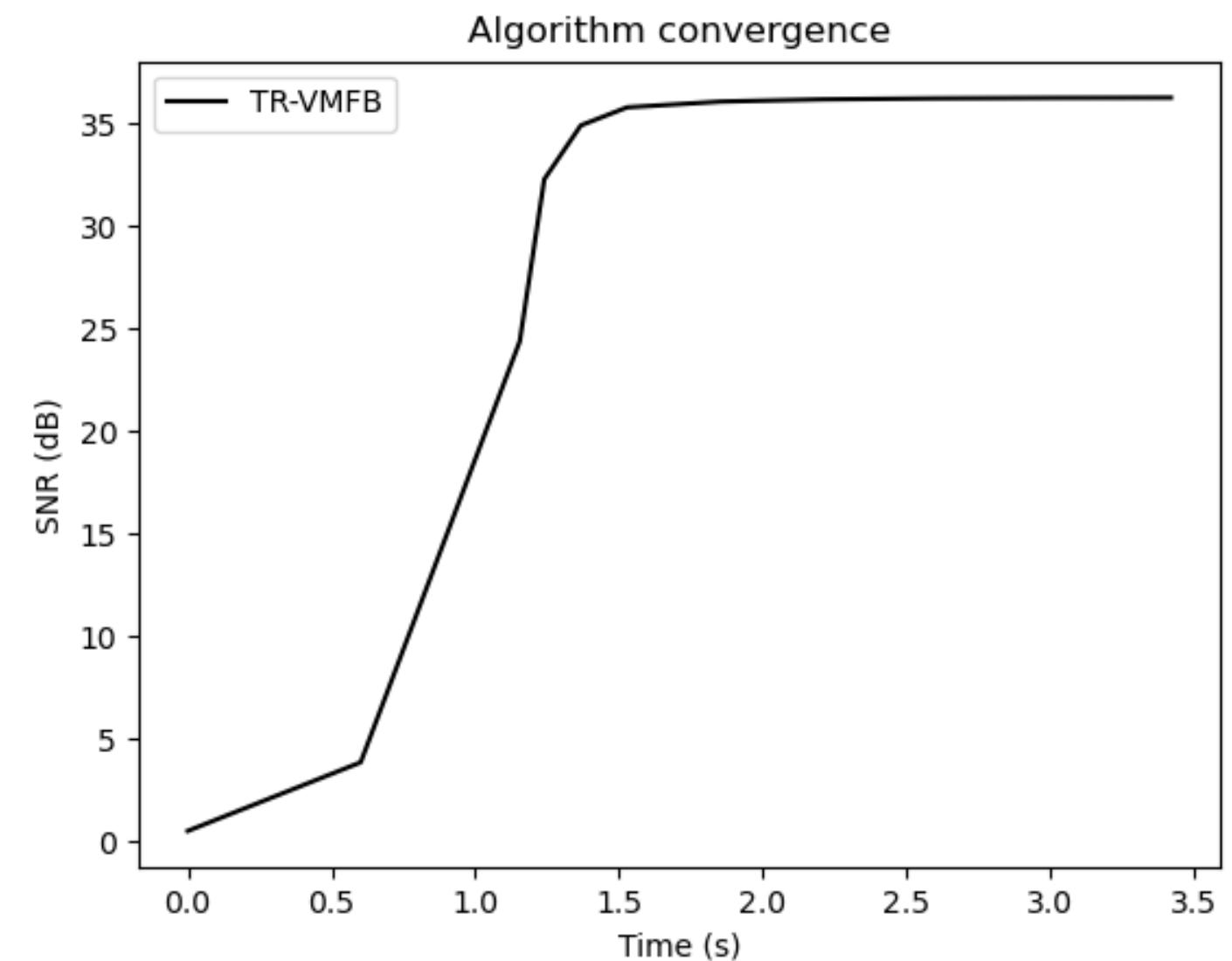
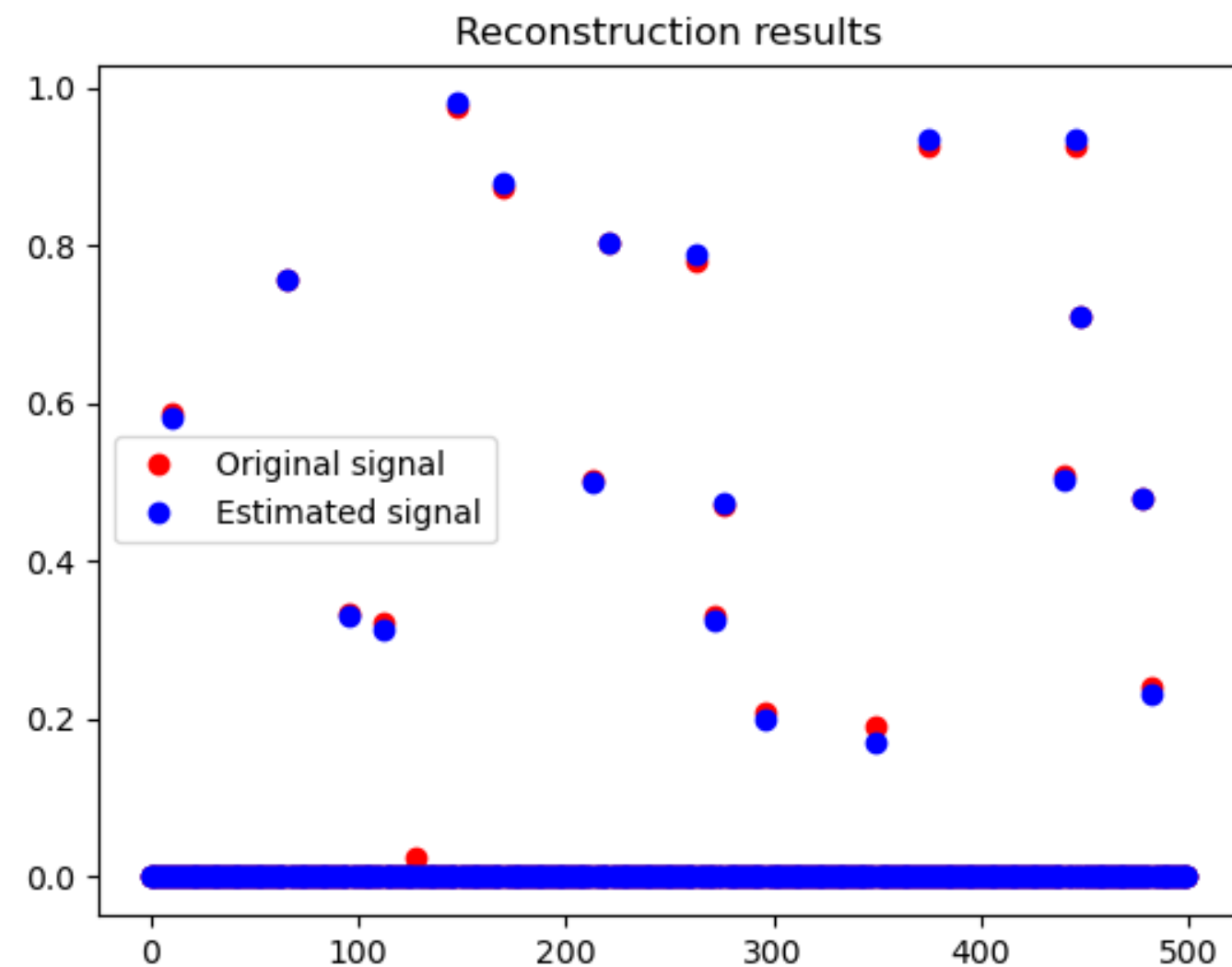


4. TESTS ET COHÉRENCE DES RÉSULTATS

Exemple d'exécution

Caractéristique du signal d'origine :

- Longueur du signal Sparse : 500
- Nombre de peak (valeur non nulle) : 20
- Metrique utilisée : FBVM-TR



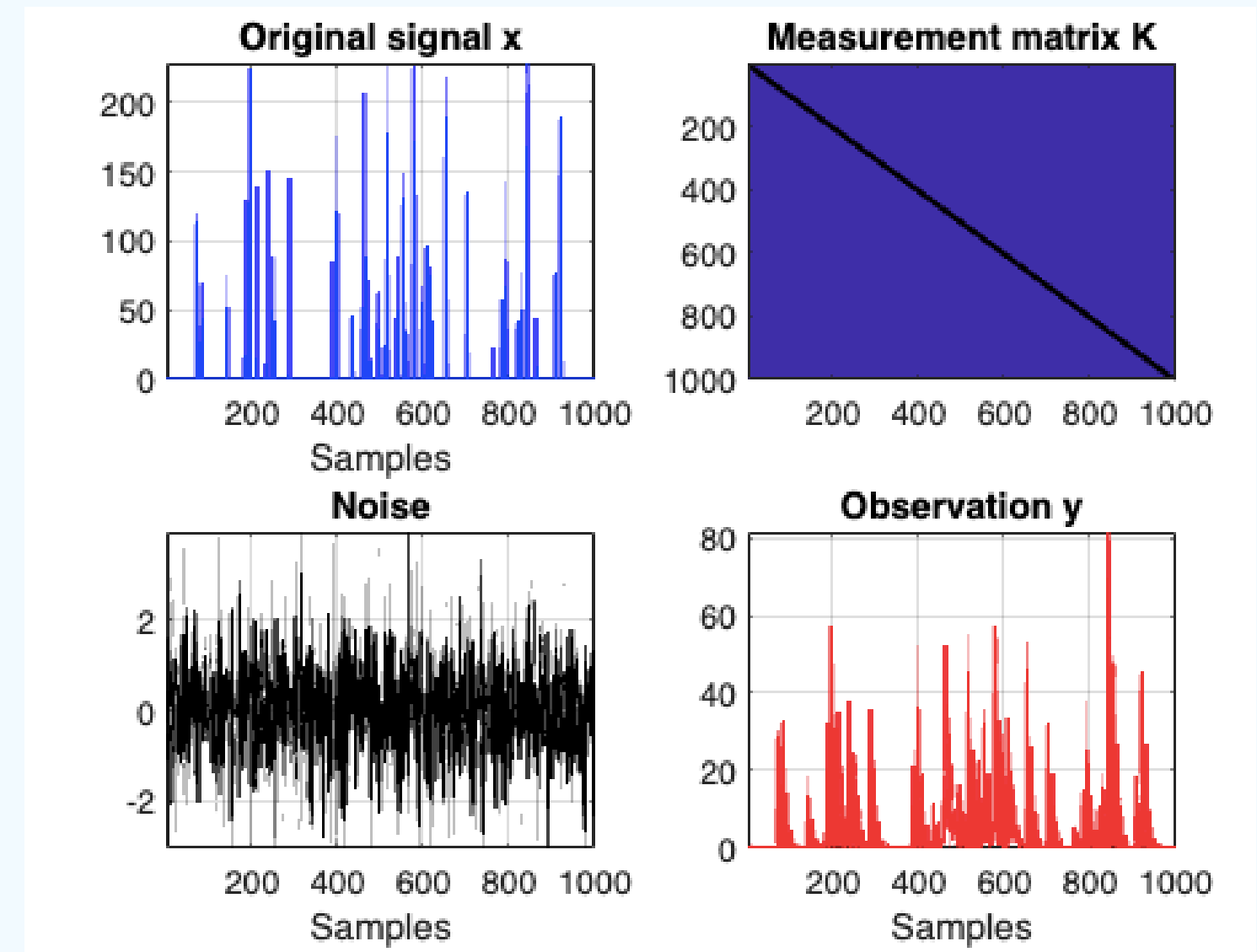
Comparaison des implémentations Matlab et Python (1) - Configuration du test et données utilisées

Paramètres utilisés

$\eta = 2E-6$
 $\alpha = 7E-7$
 $\beta = 3E-2$
 $p = 0.75$
 $q = 2$
 $\text{nbiter} = 5000$
 $\text{metric} = 2$

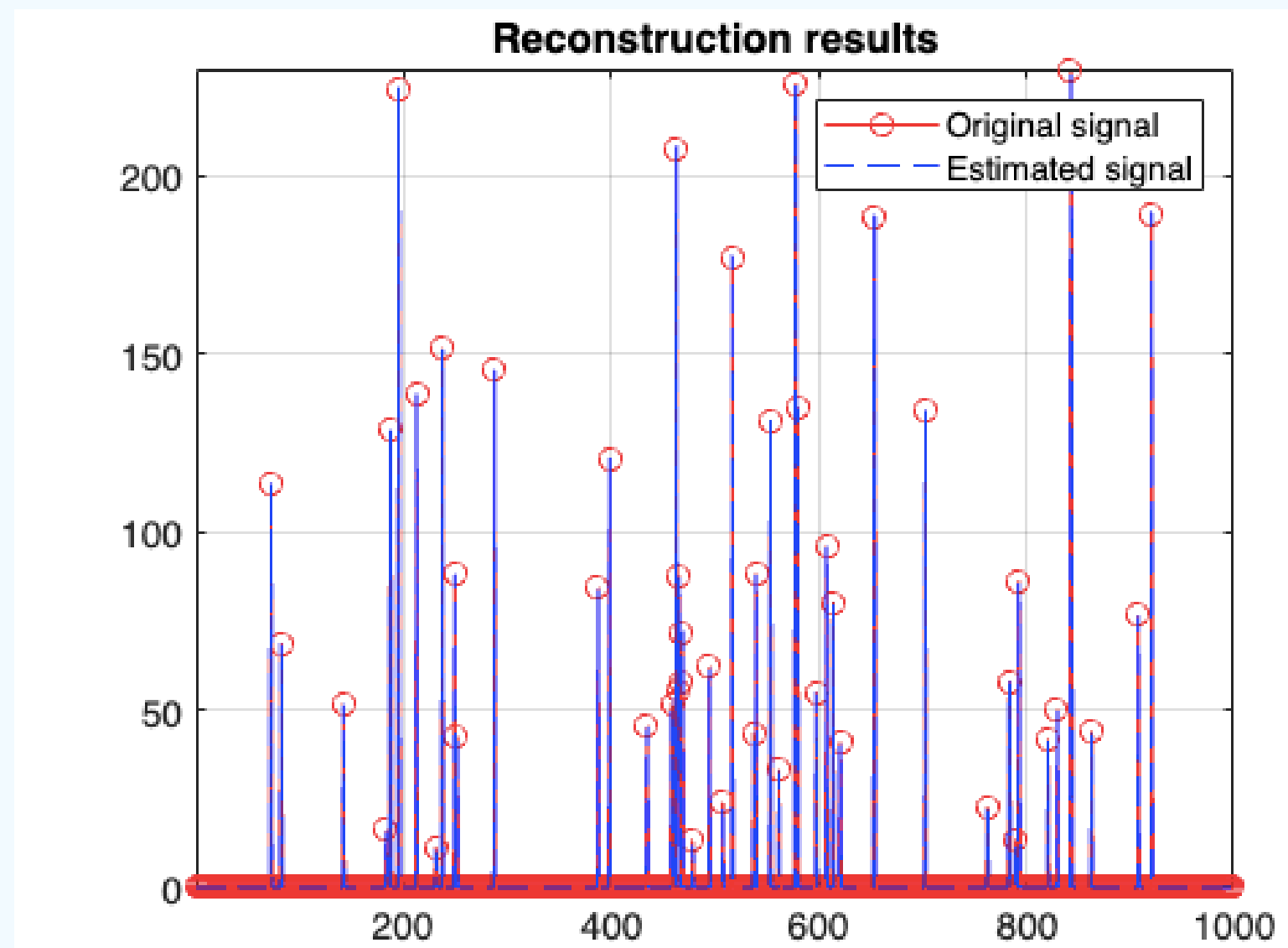
Données utilisées

Données fournies et utilisée pour obtenir les résultats du paper



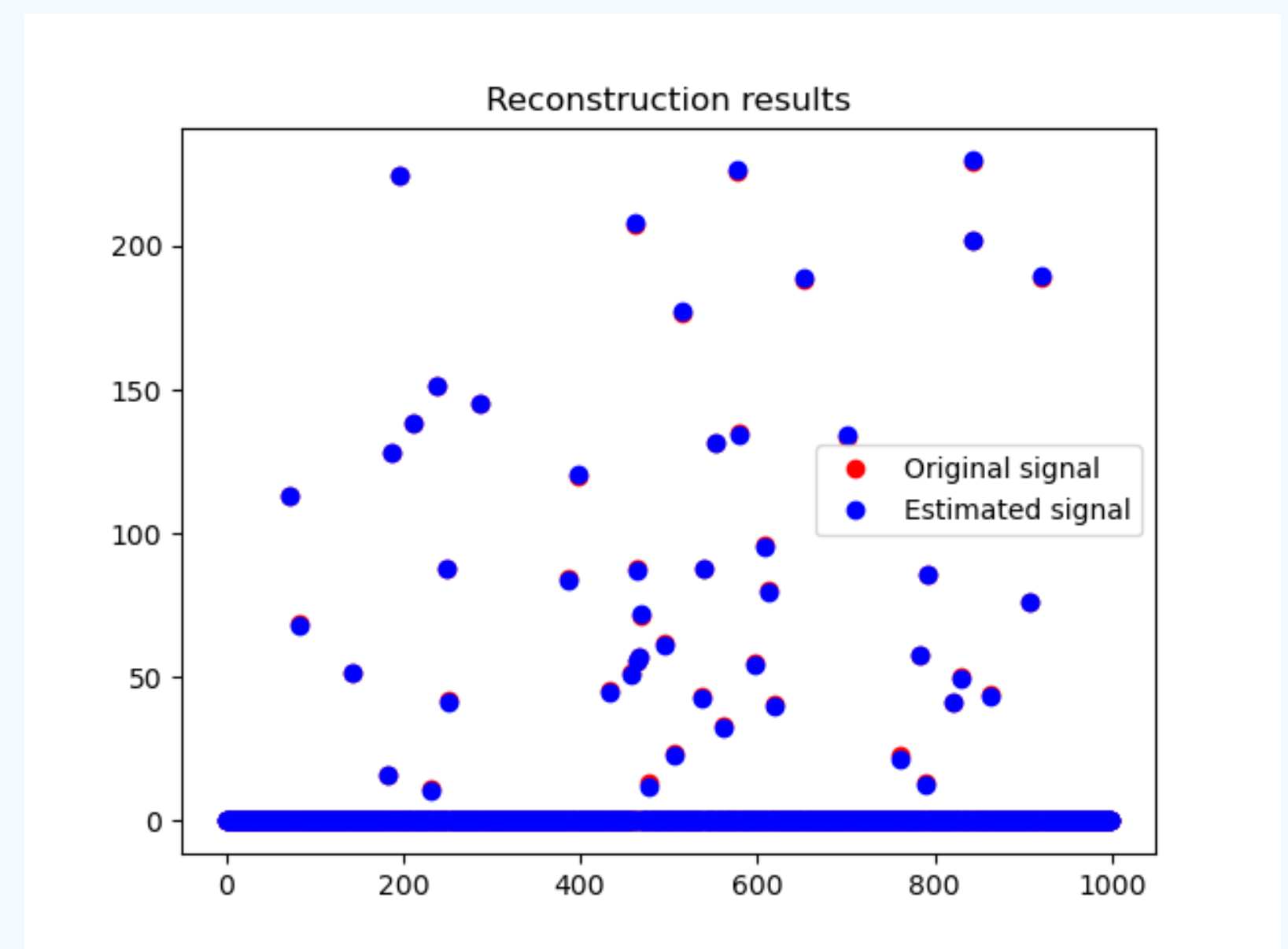
Comparaison des implémentations Matlab et Python (2) - Résultats et métriques

Implémentation Matlab



SNR = 45.8903

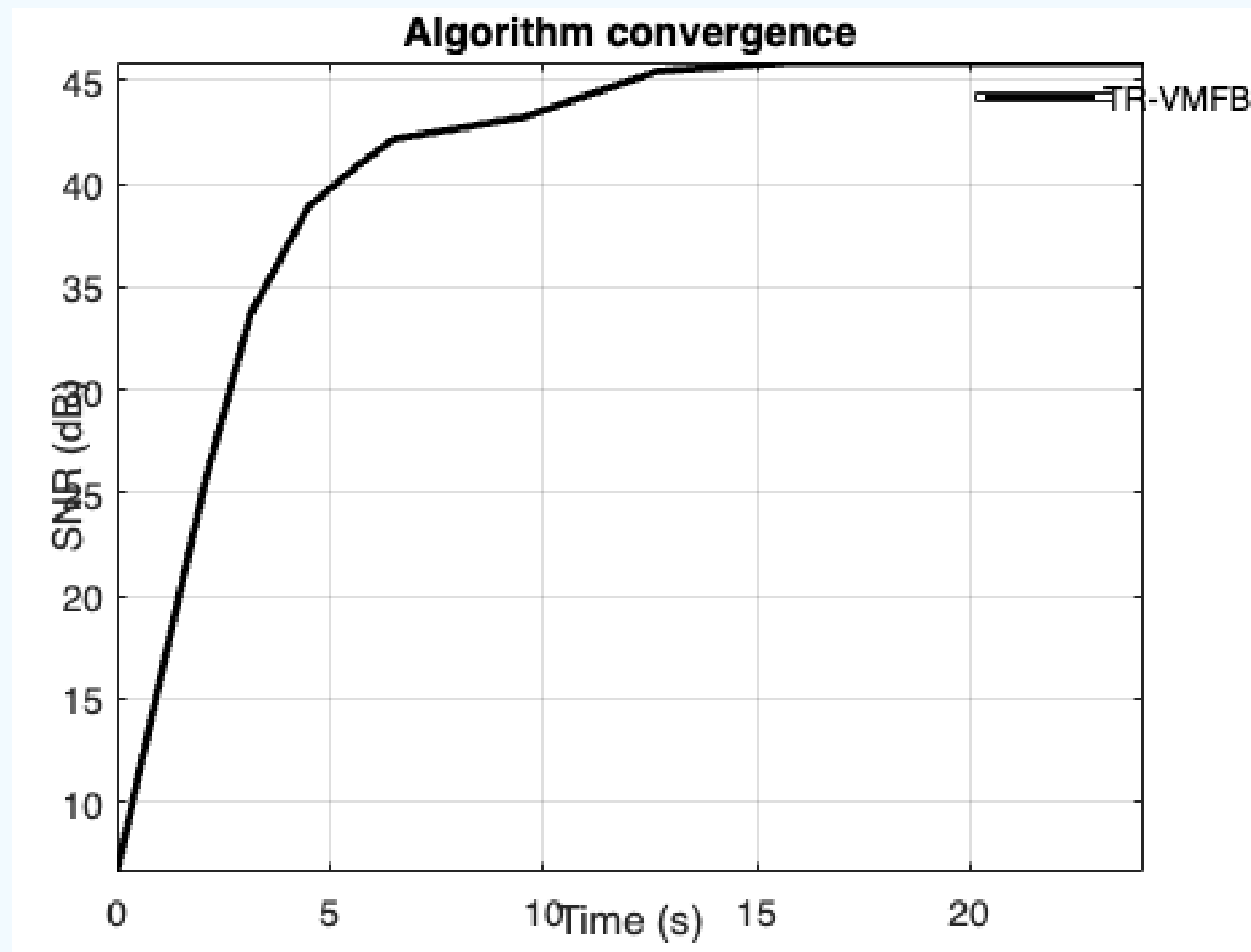
Implémentation Python



SNR = 45.8901

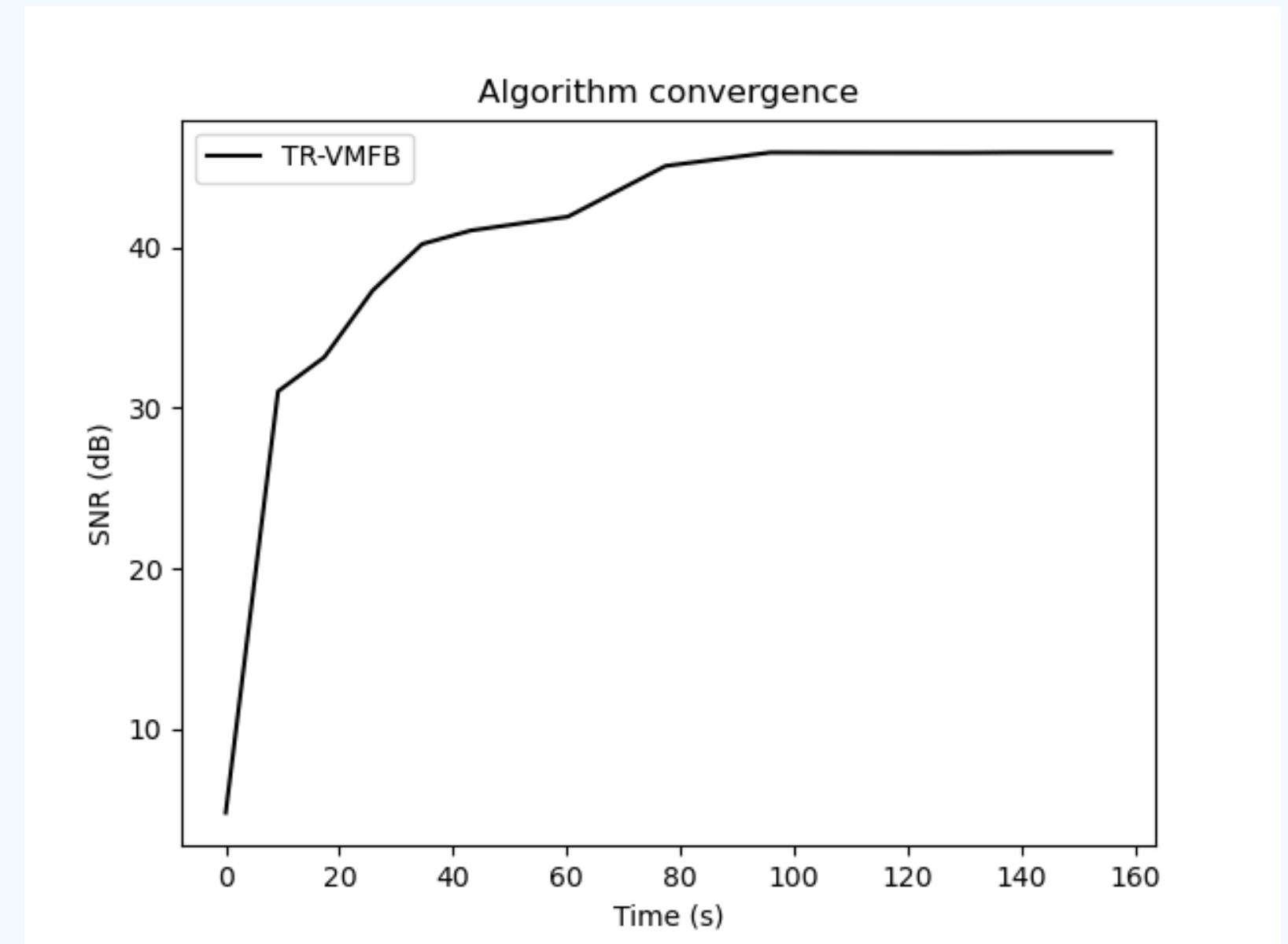
Comparaison des implémentations Matlab et Python (3) - Convergence

Implémentation Matlab



Reconstruction time is 24.0738seconds
Reconstruction in 12 iterations

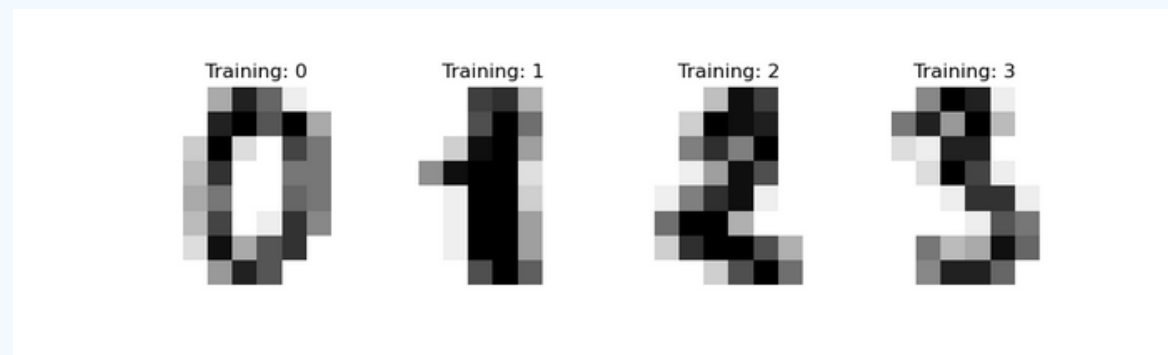
Implémentation Python



Reconstruction time is 155.8270seconds
Reconstruction in 13 iterations

Test de la fonction SPOQ en tant que pénalité de régularisation

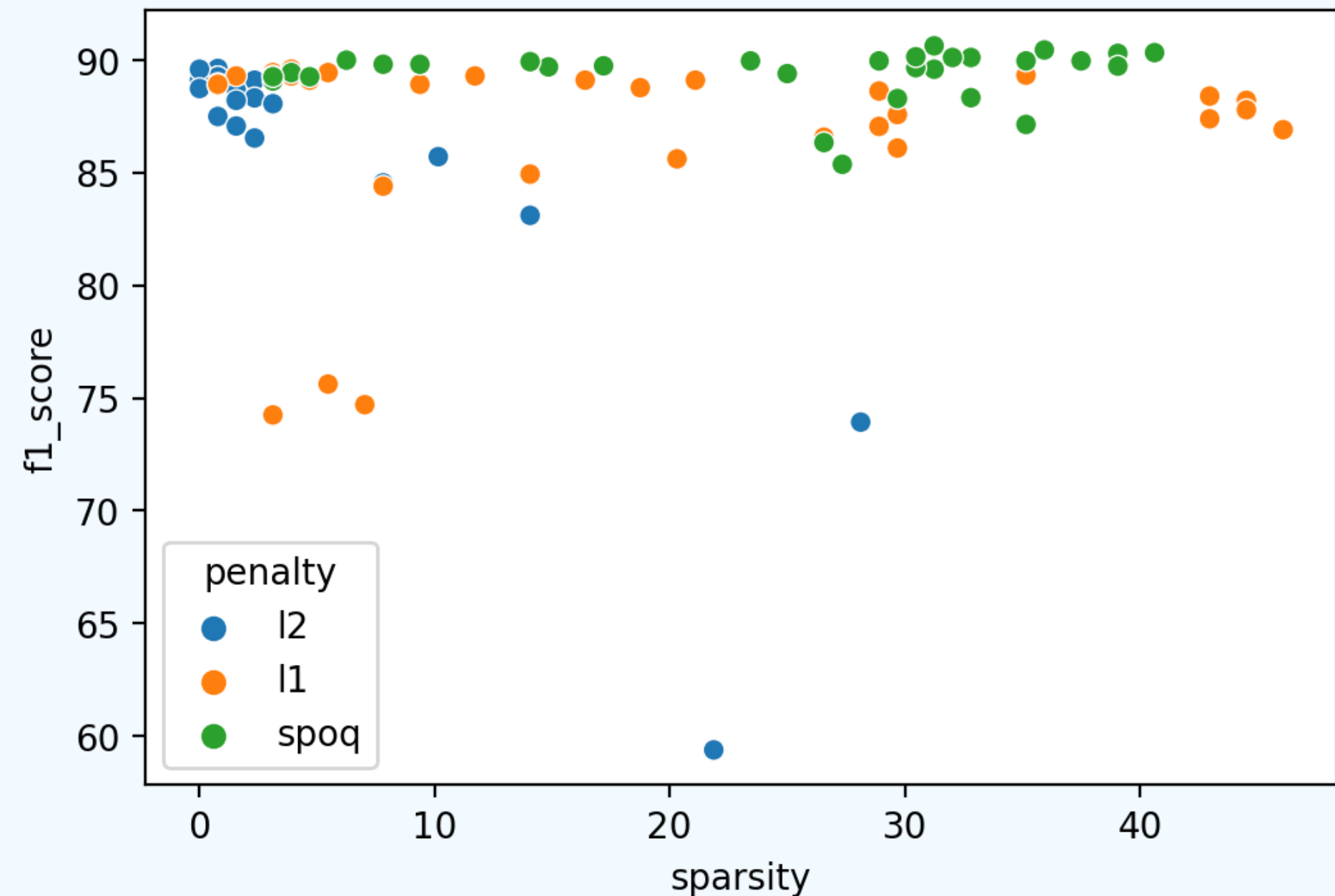
Dataset : Images 8x8 de chiffres manuscrits



Nous séparons nos données en **2 classes**
0 si $x \leq 4$ et 1 sinon

Régression logistique régularisée

Loss += λ *penalty



MERCI POUR VOTRE ATTENTION