



UNIVERSITÉ DE RENNES I
MASTER SISEA
3 Mars 2022

**TP : PROBLÈMES INVERSES
RAPPORT**

AUTEUR :

Kevin MICHALEWICZ
<kevin.michalewicz@etudiant.univ-rennes1.fr>

ENSEIGNANTS :

Prof. Di GE
Prof. Laurent ALBERA

Table des matières

1	Introduction	2
2	L'échantillonneur de Gibbs	3
3	Régularisation de Tikhonov et TV-L1	4
3.1	Algorithme MNE	4
3.2	Algorithme SISSY	10
4	Analyse de performances et comparaison des algorithmes étudiés	15
5	Conclusions	19
6	Annexe - Code MATLAB (fichier principal)	20

1 Introduction

La finalité de ce travail est de résoudre le problème inverse de la localisation des sources dans le cas d'un signal électroencéphalographique. En particulier, la plupart des dipôles seront inactifs. Par exemple, dans la Figure 1.1, ils prennent la couleur violette. Seuls les autres sont les sources actives, dans le cadre du maximum du signal d'intérêt : une pointe épileptique.

Trois approches seront exploitées : l'échantillonneur de Gibbs, l'algorithme MNE et l'algorithme SISSY. Leurs performances qualitatives et quantitatives seront comparées pour différents rapports signal/bruit et configurations de paramètres. En outre, plusieurs heuristiques seront considérées pour régler les paramètres.

Soient \mathbf{x} un vecteur d'observation $N \times 1$, \mathbf{s} un vecteur de source inconnu $D \times 1$ et \mathbf{A} une matrice de mélange $N \times D$ ($N \gg D$), le problème peut être écrit comme suit :

$$\mathbf{x} = \mathbf{As} + \mathbf{n}$$

avec \mathbf{n} un vecteur de bruit blanc gaussien de variance σ_n^2 .

Pour obtenir une solution unique, il est important d'ajouter des informations supplémentaires. Et c'est là qu'intervient la régularisation.

original source configuration: two source regions

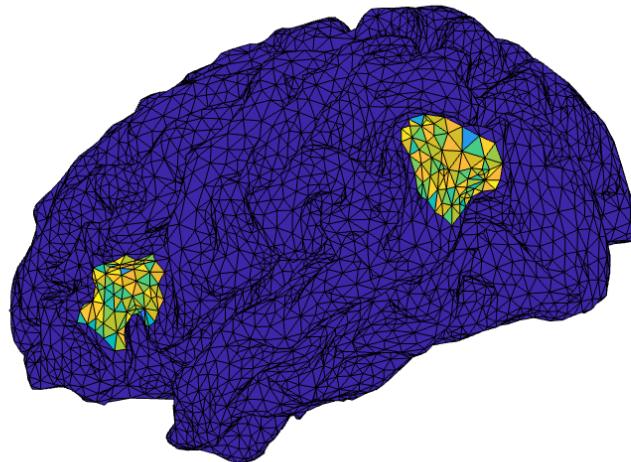


FIGURE 1.1 – Configuration de sources originale.

2 L'échantillonneur de Gibbs

L'échantillonneur de Gibbs (approche MCMC) peut être utilisé pour prélever des échantillons à partir d'une distribution conjointe, en supposant que les distributions conditionnelles complètes de chaque paramètre sont connues. Pour le problème énoncé dans le TP1, on veut estimer $\Theta = \{\mathbf{s}, \mathbf{q}, \lambda, \sigma_n^2, \sigma_s^2\}$ en sachant \mathbf{x} . La distribution *a posteriori* conjointe est

$$P(\Theta|\mathbf{x}) \propto g(\mathbf{x} - \mathbf{As}; \sigma_n^2 \mathbf{I}_N) P(\mathbf{q}; \lambda) g(\mathbf{s}; \sigma_s^2 \text{diag}(\mathbf{q})) P(\sigma_n^2) P(\sigma_s^2) P(\lambda)$$

Les expressions de certaines probabilités conditionnelles seront explicitées ci-dessous.

1. $q_i | \Theta \setminus \{q_i, s_i\}, \mathbf{x}$

En rendant explicite l'expression de la présentation de cette section et en écrivant \mathbf{A}_{-i} comme une matrice sans sa i -ième colonne et \mathbf{s}_{-i} comme un vecteur sans son i -ième élément :

$$P(q_i = 0 | \Theta \setminus \{q_i, s_i\}, \mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{x} - \mathbf{A}_{-i}\mathbf{s}_{-i}\|^2\right) (1 - \lambda)$$

$$P(q_i = 1 | \Theta \setminus \{q_i, s_i\}, \mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma_n^2} \|\mathbf{x} - \mathbf{A}_{-i}\mathbf{s}_{-i}\|^2\right) \frac{\lambda\sigma_i}{\sigma_s} \exp\left(\frac{\mu_i^2}{2\sigma_i^2}\right)$$

avec $\mu_i = \frac{\sigma_i^2}{\sigma_n^2} \mathbf{A}_i^T (\mathbf{x} - \mathbf{A}_{-i}\mathbf{s}_{-i})$ et $\sigma_i^2 = \frac{\sigma_n^2 \sigma_s^2}{\sigma_n^2 + \sigma_s^2 \|\mathbf{A}_i\|^2}$

Ensuite, et en utilisant la notation qui apparaît dans le pseudocode du TP1, on trouve que $q_i | \Theta \setminus \{q_i, s_i\}, \mathbf{x} \sim \text{Bi}\left(\frac{\nu_i}{\nu_i + 1 - \lambda}\right)$ et $\nu_i = \frac{\lambda\sigma_i}{\sigma_s} \exp\left(\frac{\mu_i^2}{2\sigma_i^2}\right)$.

2. $s_i | \Theta \setminus \{s_i\}, \mathbf{x}$

Selon le modèle B-G, si $q_i = 0$ alors $s_i = 0$. Sinon, $s_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$.

3. $\lambda | \Theta \setminus \lambda, \mathbf{x}$

En utilisant la loi *a priori* adoptée pour λ , c'est-à-dire une $\text{Be}(1, 1)$, tout simplement : $\lambda | \Theta \setminus \lambda, \mathbf{x} \sim \text{Be}(1 + L, 1 + D - L)$ avec $L = \sum_i q_i$.

Le pseudocode de l'échantillonneur de Gibbs apparaît immédiatement en utilisant les expressions précédentes et l'algorithme de base du TP1. Il a été implémenté sur MATLAB en remplissant le script *Gibbs_sampler.m* et appliqué au problème inverse de l'électroencéphalogramme (EEG). Quelques tests sont effectués pour étudier son fonctionnement.

A ce stade, on a obtenu λ en comptant les éléments non nuls de la source par rapport à ses dimensions. Pour σ_n^2 et σ_s^2 on dispose de la matrice de bruit et de la matrice des sources (respectivement). En particulier, on fait des simulations correspondant au maximum de la pointe épileptique, en faisant varier le RSB de 0.1 à 10 (Figure 2.1).

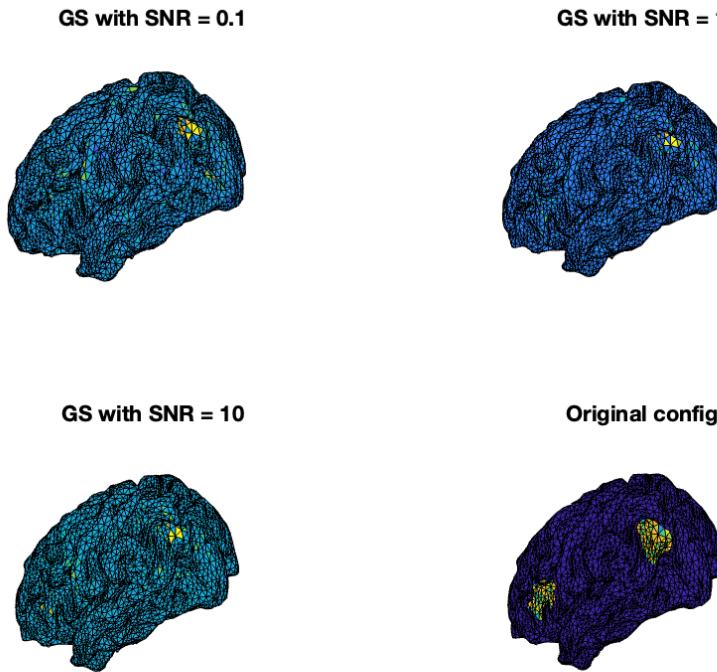


FIGURE 2.1 – Résultats de l'échantillonneur de Gibbs pour RSB entre 0.1 et 10. Comparaison avec la configuration de sources originale.

On peut dire que, globalement, les estimations ne sont pas assez proches de l'affichage original. Une des régions sources est identifiée mais l'autre est infiniment petite et seulement pour le SNR le plus élevé.

3 Régularisation de Tikhonov et TV-L1

3.1 Algorithme MNE

Tout d'abord, on montrera que la solution analytique du problème de minimisation suivant

$$\min_{\mathbf{s}} \|\mathbf{x} - \mathbf{As}\|_2^2 + \lambda \|\mathbf{s}\|_2^2$$

c'est-à-dire, avec un terme de régularisation L2 (Tikhonov), peut s'écrire sous la forme

$$\mathbf{s} = \mathbf{A}^T (\mathbf{A} \mathbf{A}^T + \lambda \mathbf{I})^{-1} \mathbf{x}$$

Dans le contexte des problèmes inverses, il est important de noter que le terme de régularisation (*prior*) est pondéré par un paramètre de régularisation λ qui permet de trouver un compromis entre l'erreur de reconstruction et l'influence du *prior*.

Si on appelle $h(\mathbf{s})$ la fonction objective, alors

$$h(\mathbf{s}) = (\mathbf{x} - \mathbf{As})^T(\mathbf{x} - \mathbf{As}) + \lambda \mathbf{s}^T \mathbf{s}$$

Les conditions nécessaires de premier ordre comportent le calcul du gradient de $h(\mathbf{s})$ et la fixation de sa valeur à zéro, ce qui donne

$$\nabla h(\mathbf{s}) = 2(-\mathbf{A}^T \mathbf{x} + \mathbf{A}^T \mathbf{As} + \lambda \mathbf{s}) = 0$$

$$\mathbf{s} = (\mathbf{AA}^T + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{x}$$

Pour arriver à la formulation souhaitée, le lemme d'inversion est utilisé avec $\mathbf{B} = \mathbf{A}$, $\mathbf{R} = \mathbf{I}$ et $\mathbf{P} = \frac{1}{\lambda} \mathbf{I}$.

$$\mathbf{s} = (\mathbf{AA}^T + \lambda \mathbf{I})^{-1} \mathbf{A}^T \mathbf{x} = \frac{1}{\lambda} \mathbf{A}^T \lambda (\mathbf{AA}^T + \lambda \mathbf{I})^{-1} \mathbf{x} = \mathbf{A}^T (\mathbf{AA}^T + \lambda \mathbf{I})^{-1} \mathbf{x}$$

Après avoir mis en œuvre cet algorithme sur MATLAB, il est évident, d'après la Figure 3.1, qu'il existe deux zones actives dont l'emplacement correspond à celui de l'affichage original ($\lambda = 1$).

En étudiant une gamme de λ allant de 0.1 à 1000, on peut dire (Figure 3.2) que progressivement, lorsque λ augmente, le dipôle actif gauche perd sa présence. En revanche, le voisinage de l'autre dipôle actif commence à gagner du terrain. En particulier, une norme L2 faible de \mathbf{s} devient préférable à la minimisation de l'erreur de reconstruction.

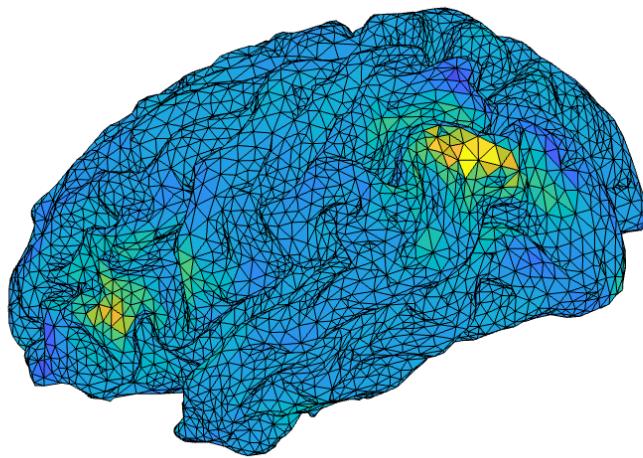
MNE output: two source regions

FIGURE 3.1 – Résultat de l'algorithme MNE avec $\lambda = 1$ et $RSB = 10$.

Si on se concentre sur le RSB (Figures 3.3, 3.4 et 3.5), pour sa valeur la plus basse, la seule reconstruction raisonnable du point de vue de la qualité est celle avec $\lambda = 1000$. Sinon, les deux sources originales ne sont pas correctement obtenues. Pour les deux autres scénarios, $\lambda = 10$ et $\lambda = 100$ semblent être de meilleures options. Dans tous les cas, un RSB plus élevé aurait tendance à donner de meilleurs résultats.

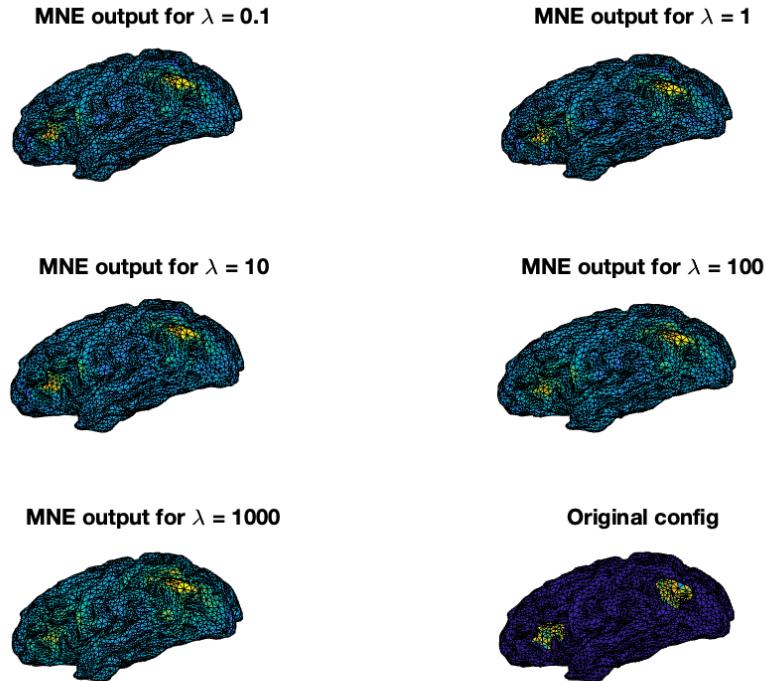


FIGURE 3.2 – Influence du paramètre de régularisation : variation de λ entre 0.1 et 1000. Comparaison avec la configuration de sources originale.



FIGURE 3.3 – Influence de λ ($\text{RSB} = 0.1$) sur la solution inverse.



FIGURE 3.4 – Influence de λ ($\text{RSB} = 1$) sur la solution inverse.

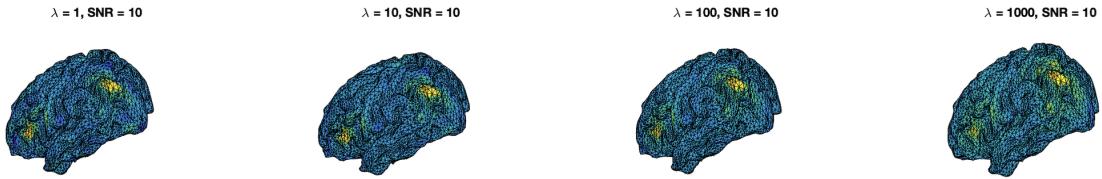


FIGURE 3.5 – Influence de λ ($RSB = 10$) sur la solution inverse.

Dans une analyse plus détaillée, trois heuristiques différentes sont considérées pour choisir le paramètre de régularisation ($RSB = 1$) : le critère du *L-curve*, le *discrepancy principle* et le *Generalized Cross Validation (GCV)*. Pour la première approche, présentée dans la Figure 3.6, la branche verticale et la branche horizontale ont été supprimées car elles correspondent à des cas extrêmes qui ne présentent pas d'intérêt : elles privilégiaient juste le premier terme ou juste le second du problème d'optimisation. Le pli de la courbe (ou, en d'autres termes, le point d'inflexion) correspond, selon ce critère, au paramètre de régularisation optimal. Dans ce cas le résultat est $\lambda = 0.01$.

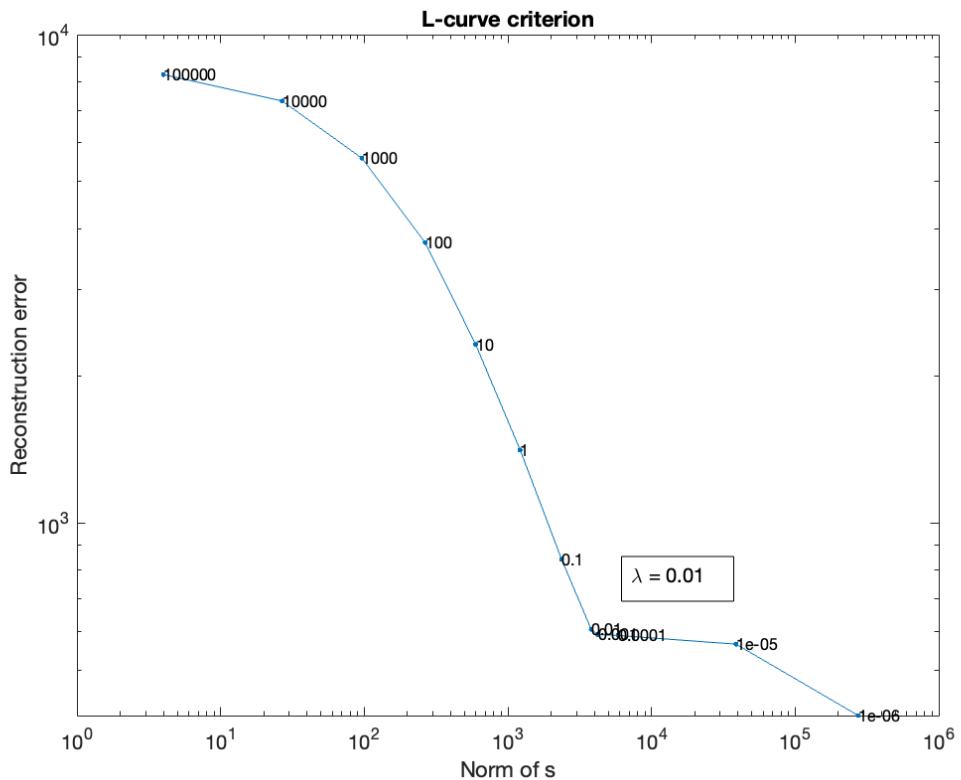


FIGURE 3.6 – L-curve pour $RSB = 1$ et λ entre 10^{-10} et 10^{10} .

Le *discrepancy principle* consiste à trouver l'intersection des courbes de l'erreur de reconstruction et de la puissance du bruit. L'abscisse correspondrait au paramètre de régularisation optimal : $\lambda = 1000$ (Figure 3.7).

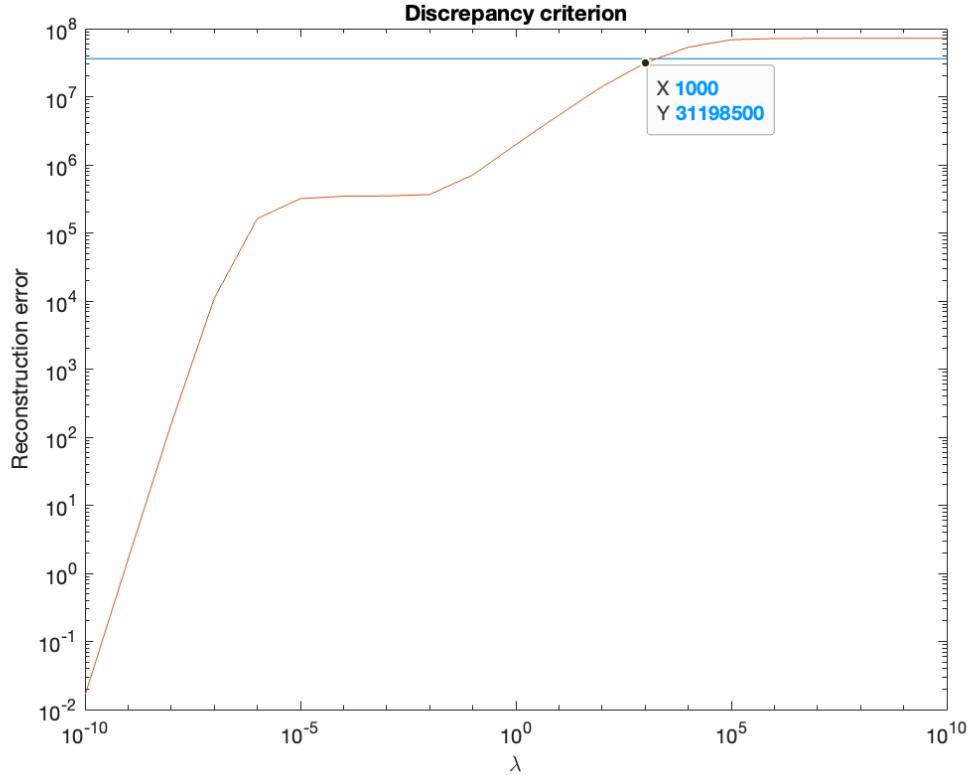


FIGURE 3.7 – Discrepancy principle pour RSB = 1 et λ entre 10^{-10} et 10^{10} .

Finalement, l'heuristique *Generalized Cross Validation (GCV)* porte sur la minimisation de la fonction suivante, en choisissant la valeur de λ associée :

$$\frac{\|\mathbf{x} - \mathbf{As}\|_2^2}{(\text{trace}\{\mathbf{I} - \mathbf{AA}^T(\mathbf{AA}^T + \lambda\mathbf{I})^{-1}\})^2}$$

Comme indiqué dans la Figure 3.8, une fois de plus, $\lambda = 1000$. Les branches horizontales ont été supprimées car elles ne présentent pas d'intérêt, selon la même logique que celle appliquée pour le critère *L-curve*.

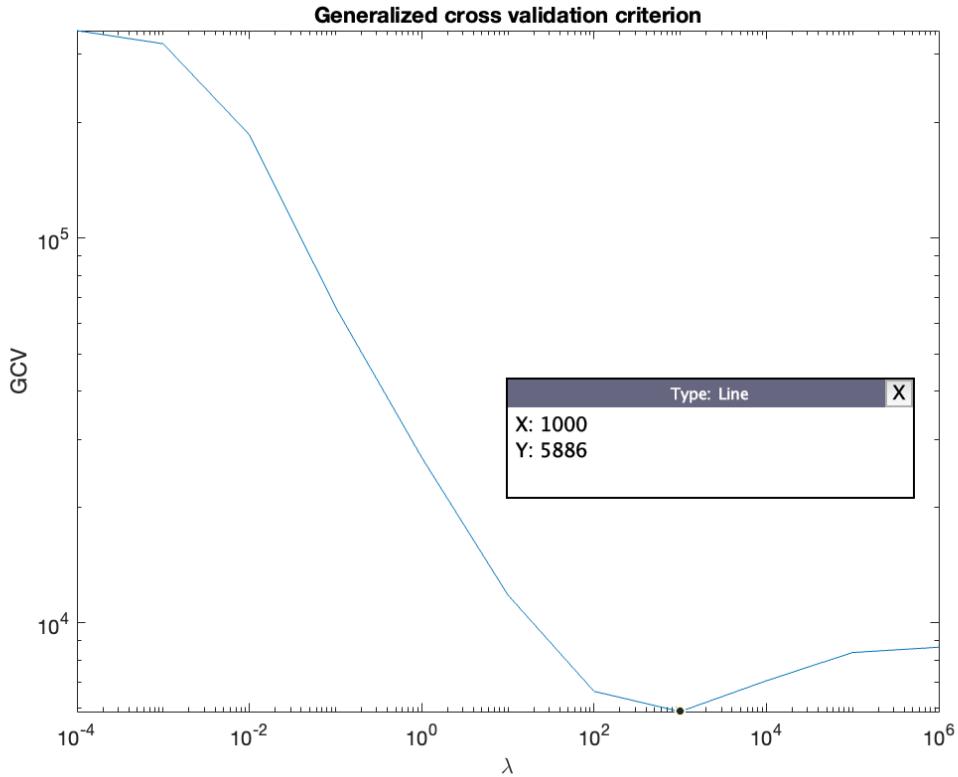


FIGURE 3.8 – GCV pour RSB = 1 et λ entre 10^{-10} et 10^{10} .

3.2 Algorithme SISSY

L'algorithme SISSY comprend un terme de régularisation basé sur la parcimonie. On peut obtenir un vecteur \mathbf{s} parcimonieux s'il contient peu d'éléments non nuls. En d'autres termes, si $\|\mathbf{s}\|_0$ (quasi-norme) est faible. Comme le problème d'optimisation associé est NP-difficile, on considère généralement une version relaxée utilisant la norme L1.

En pratique, il n'est pas toujours judicieux d'imposer la parcimonie juste dans la solution. On peut fixer cette condition après l'application d'une transformation. Pour TV-L1 cet autre terme est $\|\mathbf{T}\mathbf{s}\|_1$ avec \mathbf{T} un opérateur linéaire qui implémente le gradient sur la surface du maillage modélisant le cortex.

Maintenant, on déduira les équations de mise à jour de l'algorithme ADMM pour les variables \mathbf{s} , \mathbf{z} , \mathbf{y} et les multiplicateurs de Lagrange \mathbf{u} et \mathbf{v} , a partir du problème d'optimisation

$$\min_{\mathbf{s}} \|\mathbf{x} - \mathbf{A}\mathbf{s}\|_2^2 + \lambda(\|\mathbf{z}\|_1 + \alpha\|\mathbf{y}\|_1) \text{ t.q. } \mathbf{z} = \mathbf{T}\mathbf{s}, \mathbf{y} = \mathbf{s}$$

On ajoute quelques termes supplémentaires pour que la solution soit plus robuste et pour obtenir de meilleures propriétés de différentiabilité. Ainsi, on écrit le lagrangien augmenté

$$\begin{aligned}\mathcal{L}(\mathbf{s}, \mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{v}) = & \frac{1}{2} \|\mathbf{x} - \mathbf{As}\|_2^2 + \lambda(\|\mathbf{z}\|_1 + \alpha\|\mathbf{y}\|_1) + \mathbf{u}^T(\mathbf{z} - \mathbf{Ts}) + \mathbf{v}^T(\mathbf{y} - \mathbf{s}) \\ & + \frac{\rho}{2} \|\mathbf{z} - \mathbf{Ts}\|_2^2 + \frac{\rho}{2} \|\mathbf{y} - \mathbf{s}\|_2^2\end{aligned}$$

avec $\rho > 0$ un paramètre de pénalité. L'idée est d'initialiser les variables à zéro (car on souhaite une solution parcimonieuse) et de répéter jusqu'à convergence :

$$\mathbf{s} = \arg \min_{\mathbf{s}} \mathcal{L}(\mathbf{s}, \mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{v})$$

et de même pour les quatre autres variables. Pour commencer, on peut écrire

$$\mathcal{L}_s(\mathbf{s}) = \frac{1}{2} \|\mathbf{x} - \mathbf{As}\|_2^2 + \mathbf{u}^T(\mathbf{z} - \mathbf{Ts}) + \mathbf{v}^T(\mathbf{y} - \mathbf{s}) + \frac{\rho}{2} \|\mathbf{z} - \mathbf{Ts}\|_2^2 + \frac{\rho}{2} \|\mathbf{y} - \mathbf{s}\|_2^2$$

$$\frac{\partial \mathcal{L}_s(\mathbf{s})}{\partial \mathbf{s}} = -\mathbf{A}^T \mathbf{x} + \mathbf{A}^T \mathbf{As} - \mathbf{T}^T \mathbf{u} - \mathbf{v} - \rho \mathbf{T}^T \mathbf{z} + \rho \mathbf{T}^T \mathbf{Ts} - \rho(\mathbf{y} - \mathbf{s}) = 0$$

$$(\mathbf{A}^T \mathbf{A} + \rho(\mathbf{T}^T \mathbf{T} + \mathbf{I})) \mathbf{s} = \mathbf{A}^T \mathbf{x} + \rho \mathbf{T}^T \mathbf{z} + \mathbf{T}^T \mathbf{u} + \rho \mathbf{y} + \mathbf{v}$$

Alors, finalement

$$\mathbf{s}^{(k+1)} = (\mathbf{A}^T \mathbf{A} + \rho(\mathbf{T}^T \mathbf{T} + \mathbf{I}))^{-1} (\mathbf{A}^T \mathbf{x} + \rho \mathbf{T}^T \mathbf{z}^{(k)} + \mathbf{T}^T \mathbf{u}^{(k)} + \rho \mathbf{y}^{(k)} + \mathbf{v}^{(k)})$$

D'un autre côté,

$$\mathcal{L}_z(\mathbf{z}) = \lambda \|\mathbf{z}\|_1 + \mathbf{u}^T(\mathbf{z} - \mathbf{Ts}) + \frac{\rho}{2} \|\mathbf{z} - \mathbf{Ts}\|_2^2$$

Mais on sait que, en général, $\arg \min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 + \alpha \|\mathbf{x}\|_1 = \text{prox}_{\alpha}(\mathbf{y})$, alors si

$$\mathbf{z} = \arg \min_{\mathbf{z}} \mathcal{L}_z(\mathbf{z}) = \arg \min_{\mathbf{z}} \frac{\lambda}{\rho} \|\mathbf{z}\|_1 + \frac{1}{2} \|\mathbf{z} - (\mathbf{Ts} - \frac{\mathbf{u}}{\rho})\|_2^2$$

Donc

$$\mathbf{z}^{(k+1)} = \text{prox}_{\lambda/\rho} \left(\mathbf{Ts}^{(k+1)} - \frac{1}{\rho} \mathbf{u}^{(k)} \right)$$

En procédant de la même manière, on obtient

$$\mathbf{y}^{(k+1)} = \text{prox}_{\lambda\alpha/\rho} \left(\mathbf{s}^{(k+1)} - \frac{1}{\rho} \mathbf{v}^{(k)} \right)$$

La démarche n'est pas exactement la même pour les multiplicateurs. Par exemple, pour \mathbf{u} :

$$\mathcal{L}_u(\mathbf{u}) = \mathbf{u}^T(\mathbf{z} - \mathbf{Ts})$$

Les mises à jour sont effectuées au moyen de l'algorithme du gradient ascendant avec un pas de ρ , c'est-à-dire

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \rho (\mathbf{z}^{(k+1)} - \mathbf{T}\mathbf{s}^{(k+1)})$$

Et enfin, par analogie

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \rho (\mathbf{y}^{(k+1)} - \mathbf{s}^{(k+1)})$$

Après avoir mis en œuvre cet algorithme sur MATLAB, il est évident, d'après la Figure 3.9, qu'il existe deux zones actives dont l'emplacement correspond à celui de l'affichage original ($\lambda = 1$, $\alpha = 0.1$ et RSB = 10).

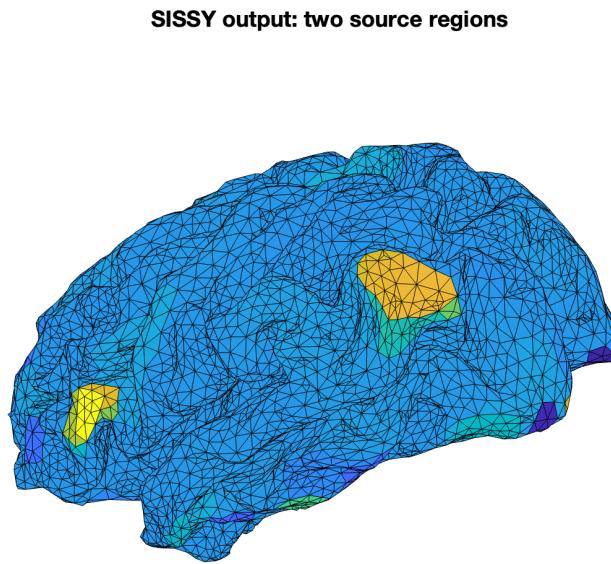


FIGURE 3.9 – Résultat de l'algorithme SISSY avec $\lambda = 1$, $\alpha = 0.1$ et RSB = 10.

Avec une valeur fixe de α et du RSB, différentes valeurs de λ ont été étudiées. Une analyse qualitative de la Figure 3.10 conduit à penser qu'une valeur proche de $\lambda = 10$ correspond au cas avec un meilleur comportement.

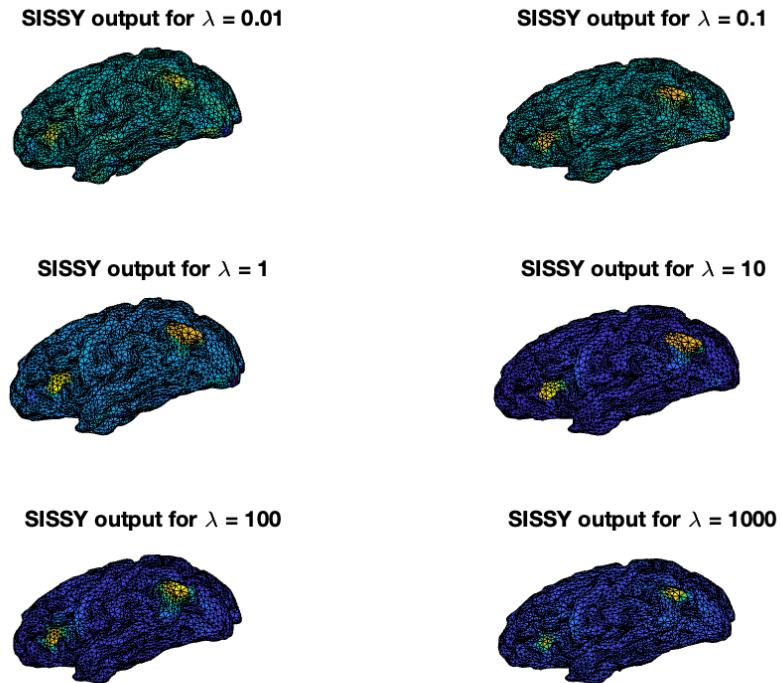


FIGURE 3.10 – Influence du paramètre de régularisation : variation de λ entre 0.01 et 1000.

Maintenant, avec une valeur fixe de $\lambda = 10$, α a été étudié entre 0 et 1 (Figure 3.11). Lorsqu'il augmente, il fait essentiellement rétrécir les régions actives car il donne un poids relatif plus élevé au deuxième partie (celle qui concerne la norme L1 du vecteur inconnu s) du terme de régularisation. Ainsi, qualitativement, une petite valeur de α serait idéale. On propose de le fixer à 0.1.

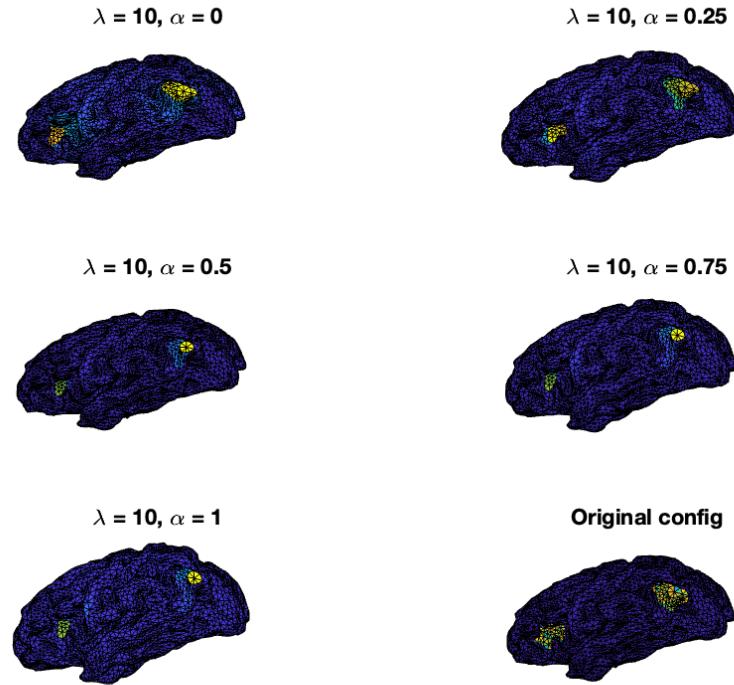


FIGURE 3.11 – Influence du paramètre α : variation entre 0 et 1 pour $\lambda = 10$. Comparaison avec la configuration de sources originale.

Une heuristique basée sur L0 a été analysée car c'est la véritable contrainte que l'on veut imposer. La valeur de λ qui la minimise peut être trouvée dans la Figure 3.12. Alors, effectivement $\lambda = 10$.

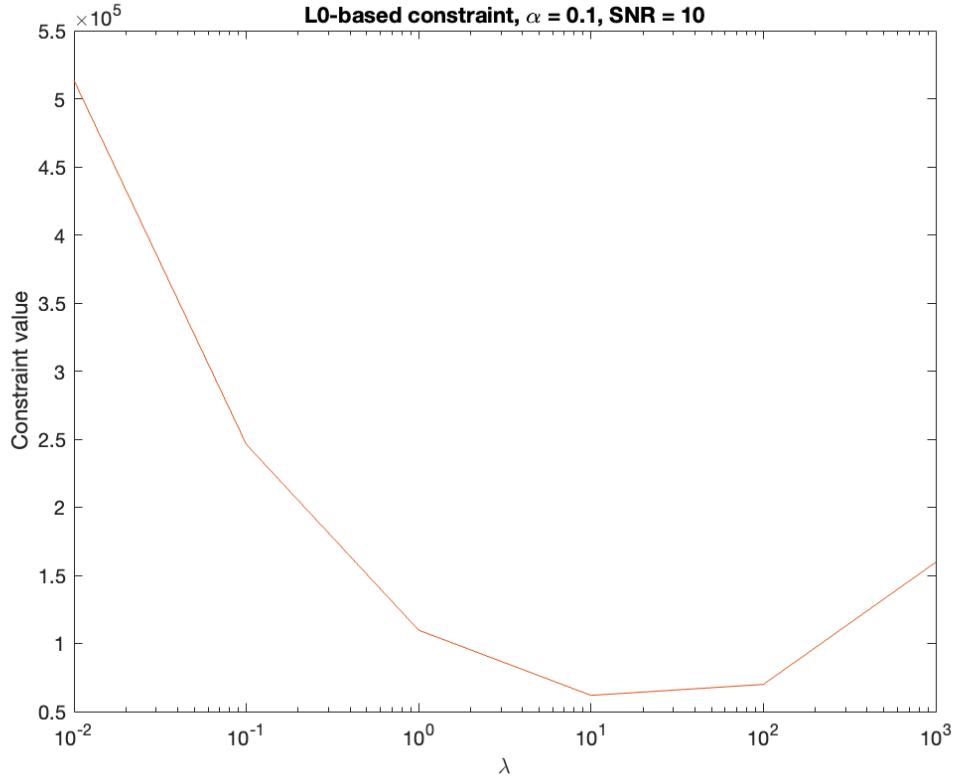


FIGURE 3.12 – Tracé pour une contrainte basée sur la norme L0 en fonction de λ

4 Analyse de performances et comparaison des algorithmes étudiés

Dans cette partie, les algorithmes précédents seront comparés entre eux et par rapport à la configuration des sources originale d'un point de vue qualitatif et quantitatif. Pour l'échantillonneur de Gibbs : $\sigma_n^2 = 3e3$, $\sigma_s^2 = 20$ et $\lambda \sim \text{Be}(1.01, 2e3)$. Pour MNE : $\lambda = 1000$. Pour SISSY : $\lambda = 10$ et $\alpha = 0.1$.

Dans les Figures 4.1, 4.2 et 4.3, la comparaison est effectuée pour trois valeurs différentes de RSB et un bruit gaussien ajouté au niveau des capteurs. On peut voir que SISSY surpassé à l'œil les autres méthodes pour $RSB = 1$ et $RSB = 10$. Cependant, pour la valeur la plus basse, les résultats peuvent être considérés comme peu satisfaisants et même surpassés par MNE qui identifie correctement le dipôle actif de droite.

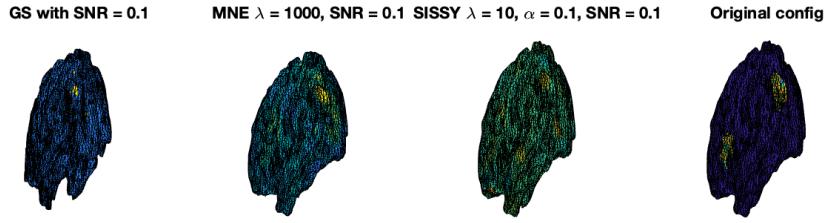


FIGURE 4.1 – Les solutions inverses des trois algorithmes pour RSB = 0.1.

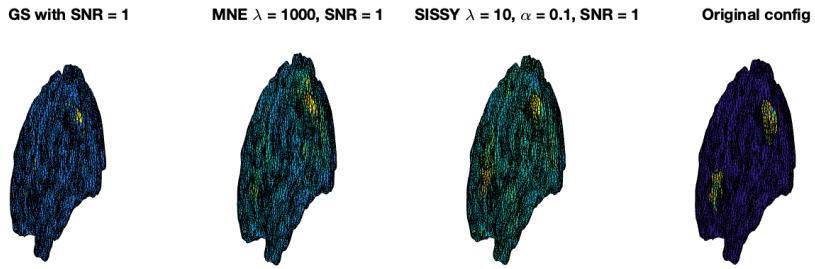


FIGURE 4.2 – Les solutions inverses des trois algorithmes pour RSB = 1.

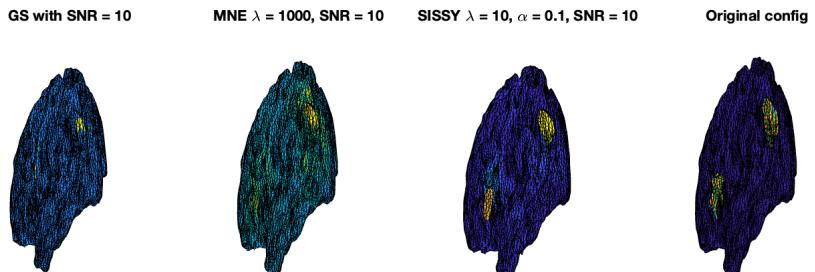


FIGURE 4.3 – Les solutions inverses des trois algorithmes pour RSB = 10.

Ensuite, on considère le cas d'un bruit spatialement corrélé. Pour cela, une matrice **Snoise** avec des éléments nuls associés aux dipôles actifs de la configuration de la source originale et des composantes tirées d'une distribution gaussienne pour le reste, est générée. D'après les Figures 4.4, 4.5 et 4.6, on peut déduire que les résultats sont meilleurs que sans le bruit gaussien spatialement corrélé, ce qui est logique car il est plus informatif que le bruit blanc. Une fois de plus, de meilleures performances sont obtenues dans le contexte de RSB plus élevés.

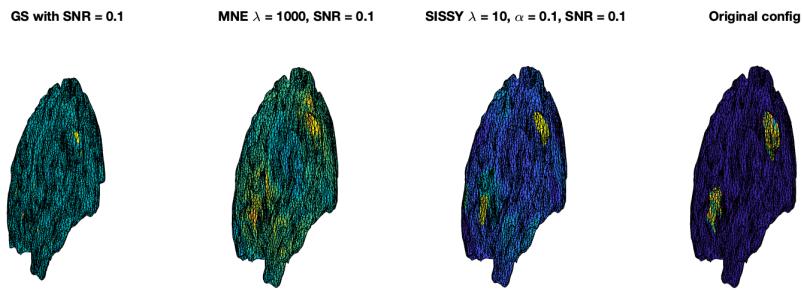


FIGURE 4.4 – Les solutions inverses des trois algorithmes pour RSB = 0.1. Bruit spatialement corrélé.

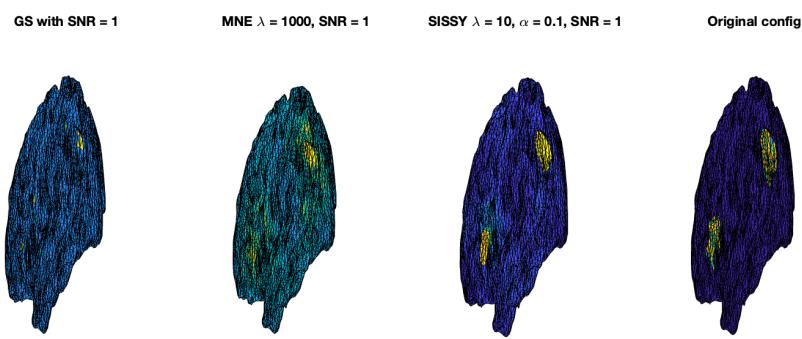


FIGURE 4.5 – Les solutions inverses des trois algorithmes pour RSB = 1. Bruit spatialement corrélé.

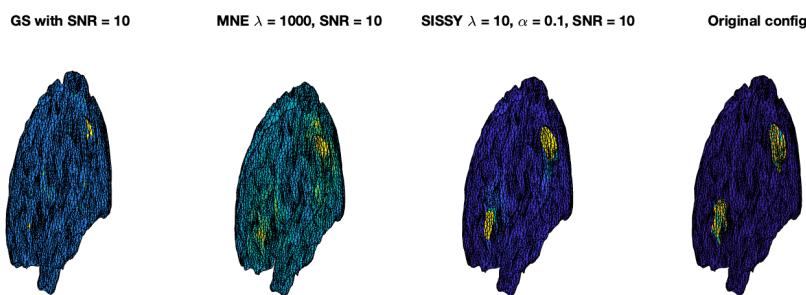


FIGURE 4.6 – Les solutions inverses des trois algorithmes pour RSB = 10. Bruit spatialement corrélé.

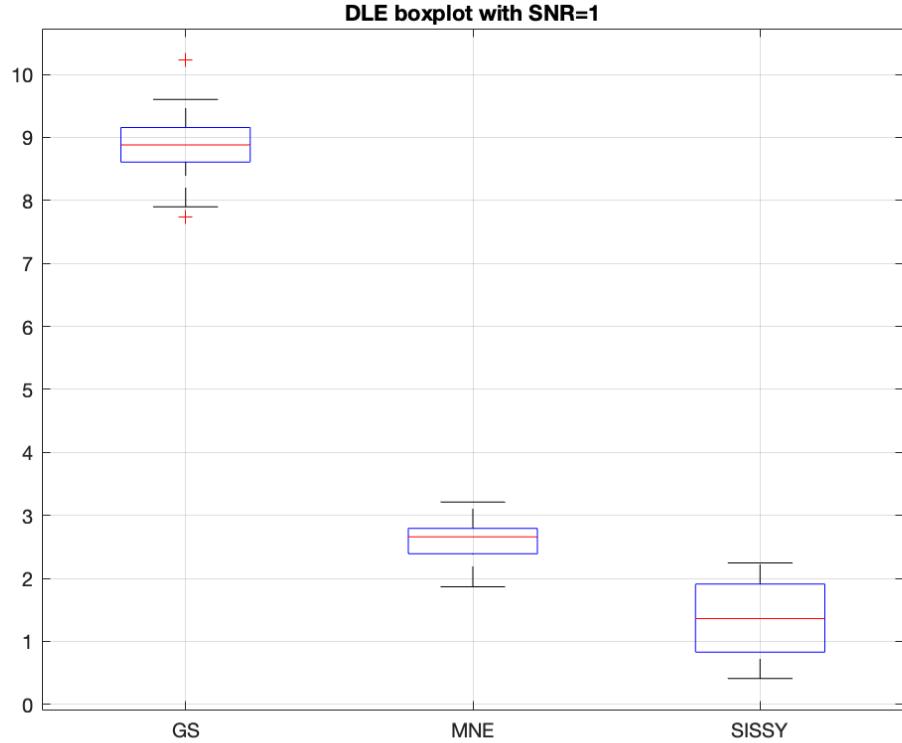


FIGURE 4.7 – Les boxplots de la DLE pour les trois algorithmes.

Pour l’aspect quantitatif, un critère d’évaluation appelé *Dipole Localization Error (DLE)* est utilisé. Tout d’abord, considérons que \mathcal{I} représente l’ensemble des indices des dipôles appartenant aux sources originales et de même pour $\hat{\mathcal{I}}$ pour les sources estimées. De plus, L et \hat{L} indiquent respectivement le nombre de dipôles actifs originaux et estimés. Ainsi, la formulation analytique est la suivante

$$\text{DLE} = \frac{1}{2L} \sum_{k \in \mathcal{I}} \min_{\ell \in \hat{\mathcal{I}}} \|\mathbf{r}_k - \mathbf{r}_\ell\| + \frac{1}{2\hat{L}} \sum_{\ell \in \hat{\mathcal{I}}} \min_{k \in \mathcal{I}} \|\mathbf{r}_k - \mathbf{r}_\ell\|$$

À l’aide de la fonction MATLAB `boxplot`, les boxplots de la DLE sont présentés pour les trois algorithmes pour $\text{RSB} = 1$ dans la Figure 4.7. On peut remarquer que MNE et SISSY ne présentent pas de *outliers* et ont les meilleurs scores.

En prenant 5 valeurs de RSB et dix réalisations dans le cas du bruit spatialement corrélé, un plot des moyennes des DLE pour MNE et SISSY en fonction du RSB a été obtenu dans la Figure 4.8. MNE est plus robuste à des rapports signal/bruit plus faibles que SISSY, comme le prouve un DLE plus élevé dans cette situation. Toutefois, on peut être sûr que SISSY sera plus performant pour des valeurs de RSB supérieures à 0.2. Tout ceci est cohérent avec les boxplots et avec l’analyse qualitative faite dans cette section.

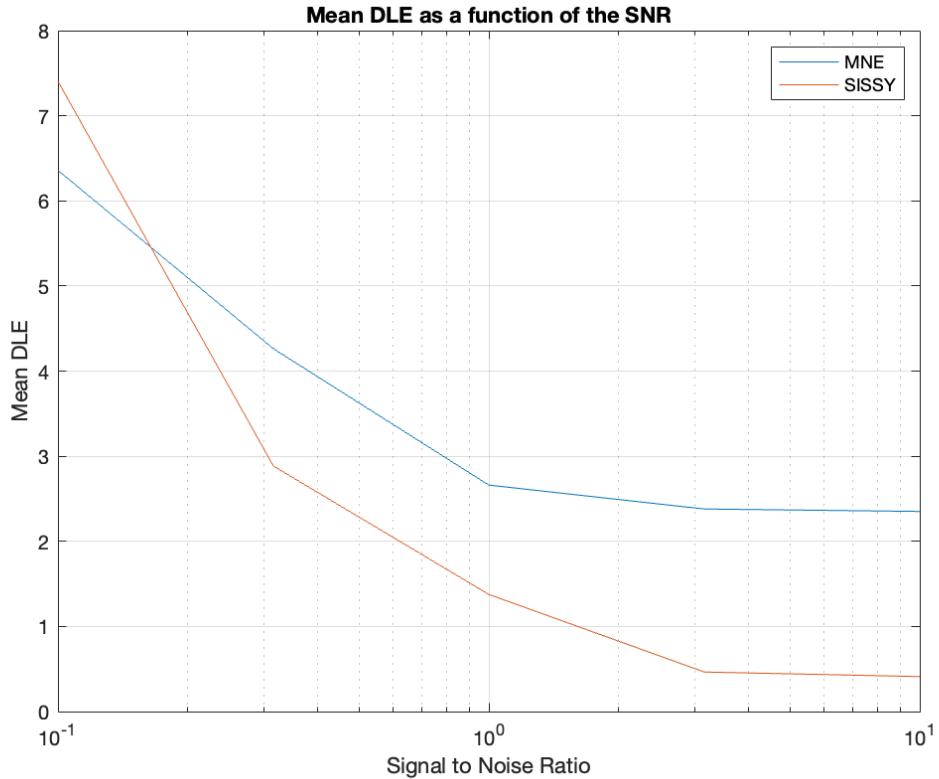


FIGURE 4.8 – La moyenne des DLE (sur 10 réalisations) en fonction du RSB pour MNE et SISSY.

5 Conclusions

En conclusion, dans ce travail, trois approches ont été proposées pour résoudre un problème inverse de localisation de sources EEG dans une pointe épileptique. Les meilleurs résultats ont été obtenus avec l'algorithme *Source Imaging Based on Structured Sparsity (SISSY)* avec $\lambda = 10$ et $\alpha = 0.1$ tant d'un point de vue qualitatif que quantitatif. En fait, l'utilisation de contraintes basées sur la parcimonie, sachant que les dipôles actifs sont très peu nombreux, est logique pour ce problème.

Il a également été déduit que l'algorithme *Minimum Norm Estimate (MNE)* est plus robuste aux mauvais RSB et que, en général, les résultats sont meilleurs lorsque le rapport signal/bruit augmente et que le bruit est spatialement corrélé.

6 Annexe - Code MATLAB (fichier principal)

```

1 clear ;
2 close all ;
3 clc ;
4
5 load TP_data ;
6
7 %generate linear mixture of source signals
8 Xs=G*S ;
9 %determine maximum of the signal of interest (here an
   epileptic spike) to
10 %apply source localization algorithms to this time point
11 [~,id]=max(mean(S,1)) ;
12
13 %visualize original source distribution
14 figure ; trisurf(mesh.f ,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
   S(:,id)) ;
15 title('original source configuration: two source regions',,
   FontSize',18) ; axis off ;
16
17 %generate Gaussian random noise
18 Noise=randn(size(Xs)) ;
19
20 %normalize noise
21 Noise=Noise/norm(Noise,'fro')*norm(Xs,'fro') ;
22
23 %signal to noise ratio
24 SNR=1 ;
25
26 %generate noisy data according to given SNR
27 X=Xs+1/sqrt(SNR)*Noise ;
28
29 %visualize data (for a reduced number of sensors whose
   indices are
30 %specified by idx_electrodes)
31 plot_eeg(X(idx_electrodes,:),max(max(X(idx_electrodes,:)))) ,
   256,channel_names) ;
32 title('noisy EEG data',FontSize',18) ;
33
34 %% Gibbs sampler
35
36 SNR=logspace(-1,1,3) ;
37
38 for i=1:length(SNR)
39     X=Xs+1/sqrt(SNR(i))*Noise ;

```

```

40 subplot(2,2,i)
41 [SOut,LambdaOut]=Gibbs_sampler(X(:,id),G);
42 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),SOut
    );
43 title('GS with SNR = '+string(SNR(i))); axis off;
44 end
45 subplot(2,2,4)
46 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),S(:,id))
    ;
47 title('Original config'); axis off;
48
49 %% MNE algorithm
50
51 lambda=1;
52 s_MNE=MNE(X,G,lambda);
53 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
    s_MNE(:,id));
54 title('MNE output: two source regions','FontSize',18); axis
    off;
55
56 %Modifying lambda - SNR=10
57
58 lambda=logspace(-1,3,5);
59
60 for i=1:length(lambda)
61     subplot(3,2,i);
62     s_MNE=MNE(X,G,lambda(i));
63     trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
        s_MNE(:,id)); axis off;
64     title('MNE output for \lambda = ' + string(lambda(i)))
        hold on
65 end
66
67
68 subplot(3,2,6)
69 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),S(:,id))
    ; axis off;
70 title('Original config')
71
72 %Modifying lambda - Each curve has a fixed SNR
73
74 lambda=logspace(0,3,4);
75 SNR=logspace(-1,1,3);
76
77 figure
78 for j=1:length(SNR)
79     for i=1:length(lambda)

```

```

80 subplot(3,4,i+4*(j-1));
81 X=Xs+1/sqrt(SNR(j))*Noise;
82 s_MNE=MNE(X,G,lambda(i));
83 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
84 s_MNE(:,id)); axis off;
85 title(['\lambda = ' + string(lambda(i)) + ', SNR = '
86 + string(SNR(j))])
87 hold on
88 end
89 % L-curve criterion
90
91 SNR=1;
92 X=Xs+1/sqrt(SNR)*Noise;
93 lambda=logspace(-10,10,21);
94
95 [error,l2_norm]=L_curve(X,G,lambda);
96
97 figure
98 l_curve_range=5:16;
99 loglog(l2_norm(l_curve_range),error(l_curve_range),'-')
100 hold on
101 text(l2_norm(l_curve_range),error(l_curve_range),string(
102 lambda(l_curve_range)),FontSize',7)
103 title('L-curve criterion');
104 xlabel('Norm of s');
105 ylabel('Reconstruction error');
106 dim=[.62 .0 .3 .3];
107 str='\lambda = '+string(lambda(9));
108 annotation('textbox',dim,'String',str,'FitBoxToText','on');
109 % Discrepancy criterion
110
111 noise_power=discrepancy_principle(Noise,lambda);
112
113 figure
114 loglog(lambda,noise_power)
115 hold on
116 loglog(lambda,error.^2)
117 title('Discrepancy criterion');
118 xlabel('\lambda');
119 ylabel('Reconstruction error');
120
121 % Generalized cross validation
122
```

```

123 GCV=generalized_cross_validation(X,G,lambda);
124 GCV_range=7:17;
125 figure
126 loglog(lambda(GCV_range),GCV(GCV_range))
127 hold on
128 title('Generalized cross validation criterion');
129 xlabel('\lambda');
130 ylabel('GCV');

131 %% SISSY algorithm
133
134 SNR=10;
135 X=Xs+1/sqrt(SNR)*Noise;
136 lambda=1; alpha=0.1;
137 s_SISSL=SISSL(X,G,variation_operator(mesh,'face'),lambda,
138 alpha);
139 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
140 s_SISSL(:,id));
141 title('SISSL output: two source regions','FontSize',18);
142 axis off;

143 %Modifying lambda - SNR=10
144
145 for i=1:length(lambda)
146 subplot(3,2,i);
147 s_SISSL=SISSL(X,G,variation_operator(mesh,'face'),
148 lambda(i),alpha);
149 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
150 s_SISSL(:,id)); axis off;
151 title('SISSL output for \lambda = ' + string(lambda(i)))
152 hold on
153 end

154 %Modifying alpha - lambda=10
155
156 alpha=linspace(0,1,5);
157 lambda=10;
158 for i=1:length(alpha)
159 subplot(3,2,i);
160 s_SISSL=SISSL(X,G,variation_operator(mesh,'face'),
161 lambda,alpha(i));
162 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3)),

```

```

162     s_SISSLY(:,id)); axis off;
163     title('lambda = '+string(lambda)+', alpha = ' +
164         string(alpha(i)))
165     hold on
166 end
167 subplot(3,2,6)
168 trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),S(:,id))
169     ; axis off;
170     title('Original config')
171
172 % Finding lambda using the L0 norm
173
174 alpha=0.1;
175 lambda=logspace(-2,3,6);
176
177 for i=1:length(lambda)
178     s_SISSLY=SISSLY(X,G,variation_operator(mesh,'face'),
179         lambda(i),alpha);
180     thr=0.01*max(max(s_SISSLY));
181     s_thr=s_SISSLY;
182     s_thr(find(abs(s_thr)<abs(thr)))=0;
183     Ts_thr=variation_operator(mesh,'face')*s_SISSLY;
184     thr=0.01*max(max(Ts_thr));
185     Ts_thr(find(abs(Ts_thr)<abs(thr)))=0;
186
187     constraint(i)=nnz(Ts_thr)+alpha*nnz(s_thr);
188 end
189
190 semilogx(lambda,constraint)
191 hold on
192 title('L0-based constraint, alpha = '+string(alpha)+', SNR =
193     '+string(SNR));
194 xlabel('lambda');
195 ylabel('Constraint value');
196
197 %% Qualitative comparison
198
199 % Non-correlated noise
200
201 SNR=logspace(-1,1,3);
202 lambda_MNE=1000;
203 lambda_SISSLY=10;
204 alpha=0.1;
205
206 for i=1:length(SNR)
207     X=Xs+1/sqrt(SNR(i))*Noise;

```

```

203 figure
204 subplot(1,4,1)
205 [SOut,LambdaOut] = Gibbs_sampler(X(:, id ),G);
206 trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,SOut
    );
207 title('GS with SNR = '+string(SNR(i))); axis off;
208 hold on
209 subplot(1,4,2)
210 s_MNE=MNE(X,G,lambda_MNE);
211 trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,
    s_MNE(:, id )); axis off;
212 title('MNE \lambda = '+string(lambda_MNE)+', SNR = '+
    string(SNR(i)))
213 subplot(1,4,3)
214 s_SISSLY=SISSY(X,G,variation_operator(mesh,'face'),
    lambda_SISSLY,alpha);
215 trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,
    s_SISSLY(:, id )); axis off;
216 title('SISSY \lambda = '+string(lambda_SISSLY)+', \alpha =
    '+string(alpha)+', SNR = '+ string(SNR(i)))
217 subplot(1,4,4)
218 trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,S(:, ,
    id ));
219 title('Original config'); axis off;
220 end

221 % Correlated noise
222
223
224 Snoise=zeros(size(S));
225 Snoise(find(S==0))=randn(size(find(S==0)));
226 Noise=G*Snoise;
227 Noise=Noise/norm(Noise,'fro')*norm(Xs,'fro');

228 for i=1:length(SNR)
229     X=Xs+1/sqrt(SNR(i))*Noise;
230     figure
231     subplot(1,4,1)
232     [SOut,LambdaOut] = Gibbs_sampler(X(:, id ),G);
233     trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,SOut
        );
234     title('GS with SNR = '+string(SNR(i))); axis off;
235     hold on
236     subplot(1,4,2)
237     s_MNE=MNE(X,G,lambda_MNE);
238     trisurf(mesh.f ,mesh.v(:,1) ,mesh.v(:,2) ,mesh.v(:,3) ,
        s_MNE(:, id )); axis off;

```

```

240     title( 'MNE \lambda = '+string(lambda_MNE)+', SNR = '+
241         string(SNR(i)))
242     subplot(1,4,3)
243     s_SISSLY=SISSLY(X,G,variation_operator(mesh,'face'),%
244         lambda_SISSLY,alpha);
245     trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),%
246         s_SISSLY(:,id)); axis off;
247     title('SISSLY \lambda = '+string(lambda_SISSLY)+', \alpha %
248         = '+string(alpha)+', SNR = '+ string(SNR(i)))
249     subplot(1,4,4)
250     trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),S(:,%
251         id));
252     title('Original config'); axis off;
253 end
254
255 %% Quantitative comparison
256
257 % Identifying positions of the original source
258 % configuration
259
260 r_grid=zeros(size(mesh.f));
261 k=1;
262 thr=1;
263
264 while(k<=size(mesh.f,1))
265     r_grid(k,:)= mesh.v(mesh.f(k,:));
266     k=k+1;
267 end
268 original=zeros(size(S(:,1)));
269 inactive=find(S==0);
270 active=unique(find(abs(S(:,id))>thr));
271
272 original(active)=1;
273 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),%
274         original);
275 title('Original config'); axis off;
276
277 % DLE matrix for each algorithm, SNR=1, 10 sets, spatially
278 % correlated noise
279
280 DLE_matrix=zeros(10,3);
281 SNR=1;
282 lambda_MNE=200;
283 lambda_SISSLY=10;
284 alpha=0.1;

```

```

278 for it=1:10
279
280 Snoise=zeros( size(S) );
281 Snoise( find(S==0) )=randn( size( find(S==0) ) );
282 Noise=G*Snoise;
283 Noise=Noise/norm(Noise, 'fro')*norm(Xs, 'fro');
284
285 X=Xs+1/sqrt(SNR)*Noise;
286
287 [s_GS,LambdaOut]=Gibbs_sampler(X(:,id),G);
288 active_GS=unique( find( abs(s_GS)>thr ),size(s_GS,1));
289
290 s_MNE=MNE(X,G,lambda_MNE);
291 active_MNE=unique( find( abs(s_MNE(:,id))>thr ) );
292
293 s_SISSLY=SISSLY(X,G,variation_operator(mesh, 'face'),
294 lambda_SISSLY, alpha);
295 active_SISSLY=unique( find( abs(s_SISSLY(:,id))>thr ) );
296
297 DLE_matrix(it,1)=DLE(active,active_GS,r_grid);
298 DLE_matrix(it,2)=DLE(active,active_MNE,r_grid);
299 DLE_matrix(it,3)=DLE(active,active_SISSLY,r_grid);
300
301 end
302
303 boxplot(DLE_matrix, 'Labels', { 'GS' , 'MNE' , 'SISSLY' });
304 title('DLE boxplot with SNR=1');
305 grid on;
306
307 GS_config=zeros( size(S(:,1)) );
308 GS_config(active_GS)=1;
309 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
310 GS_config);
311 title('GS'); axis off;
312
313 MNE_config=zeros( size(S(:,1)) );
314 MNE_config(active_MNE)=1;
315 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
316 MNE_config);
317 title('MNE'); axis off;
318
319 SISSLY_config=zeros( size(S(:,1)) );
320 SISSLY_config(active_SISSLY)=1;
321 figure; trisurf(mesh.f,mesh.v(:,1),mesh.v(:,2),mesh.v(:,3),
322 SISSLY_config);
323 title('SISSLY'); axis off;

```

```

320
321 % DLE for MNE and SISSY , different SNR values , 10 sets
322
323 SNR=logspace(-1,1,5);
324 DLE_matrix=zeros(length(SNR),2);
325
326 for it=1:10
327     for j=1:length(SNR)
328
329         X=Xs+1/sqrt(SNR(j))*Noise;
330
331         s_MNE=MNE(X,G,lambda_MNE);
332         active_MNE=unique(find(abs(s_MNE(:,id))>thr));
333
334         s_SISSL=SISSL(X,G,variation_operator(mesh,'face'),
335                         lambda_SISSL,alpha);
336         active_SISSL=unique(find(abs(s_SISSL(:,id))>thr));
337         active_SISSL(find(active_SISSL==0))=size(
338             active_SISSL,1);
339
340         DLE_matrix(j,1)=DLE_matrix(j,1)+DLE(active,
341                                         active_MNE,r_grid)/10;
342         DLE_matrix(j,2)=DLE_matrix(j,2)+DLE(active,
343                                         active_SISSL,r_grid)/10;
344
345     end
346
347 end
348
349 semilogx(SNR,DLE_matrix(:,1));
350 hold on
351 semilogx(SNR,DLE_matrix(:,2));
352 xlabel('Signal to Noise Ratio');
353 ylabel('Mean DLE');
354 legend('MNE','SISSL');
355 title('Mean DLE as a function of the SNR');
356 grid on;

```