Méthodes Signal Avancées
Soustraction Adaptative de Bruit

Abdelhamied OMAR YOUSIF

Introduction

Ce TP porte sur l'application des méthodes de soustraction adaptative de bruit, en particulier à travers l'algorithme RLS (Recursive Least Squares) et l'algorithme NLMS (Normalized Least Mean Squares). L'objectif est de tester ces algorithmes dans des scénarios d'annulation de bruit et d'évaluation de leurs performances selon différents paramètres.

Partie 1 : Algorithme RLS

1. Algorithme RLS

- $d(n)$: le signal de référence ou désiré à l'instant n.
- $x(n)$: le vecteur de signal d'entrée à l'instant n.
- $w(n)$: le vecteur des poids du filtre à l'instant n.
- $P(n)$: la matrice d'inversion de la covariance à l'instant n.
- λ : le facteur d'oubli, $0 < \lambda \leq 1$.

1. Erreur d'estimation :

$$e(n) = d(n) - w(n)^T x(n) \tag{1}$$

où $e(n)$ est l'erreur entre la sortie désirée $d(n)$ et la sortie estimée $w(n)^T x(n)$.

2. Calcul du gain de filtrage $k(n)$:

$$k(n) = \frac{P(n-1)x(n)}{\lambda + x(n)^T P(n-1)x(n)} \tag{2}$$

Le vecteur $k(n)$ est appelé *gain de Kalman*. Il ajuste la contribution de l'erreur actuelle pour mettre à jour les coefficients du filtre.

3. Mise à jour des poids du filtre :

$$w(n) = w(n-1) + k(n)e(n) \tag{3}$$

Cette équation met à jour les coefficients du filtre $w(n)$ en utilisant le gain $k(n)$ et l'erreur $e(n)$.

4. Mise à jour de la matrice d'inversion de la covariance $P(n)$:

$$P(n) = \frac{1}{\lambda}\left(P(n-1) - k(n)x(n)^T P(n-1)\right) \tag{4}$$

Cette étape ajuste la matrice $P(n)$ en tenant compte du facteur d'oubli λ, permettant ainsi à l'algorithme RLS de "oublier" progressivement les échantillons passés lorsque λ est proche de 1.

2. Mise en œuvre et validation de l'algorithme RLS

Dans cette section, nous avons mis en place l'algorithme RLS sous MATLAB pour adapter un filtre de Réponse Impulsionnelle Finie (RIF) à partir des signaux x (bruit blanc) de longueur **N = 1000** et **d** (signal filtré avec un filtre RIF inconnu **h = [1, 0.8, −0.3, 0.4]**).

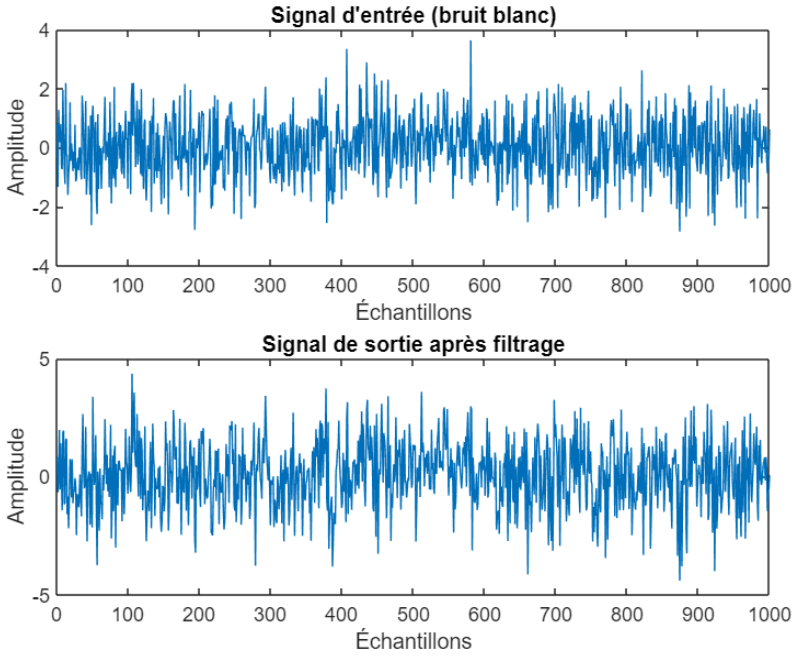

Figure 1 *: Signal d'entrée et filtré*

Pour valider l'algorithme, nous avons tracé l'erreur en fonction du temps et comparé les coefficients estimés aux valeurs réelles. Le filtre adaptatif est ajusté à chaque itération pour minimiser l'erreur entre le signal désiré d et la sortie estimée y (fin de l'algorithme l'erreur tend vers 0).

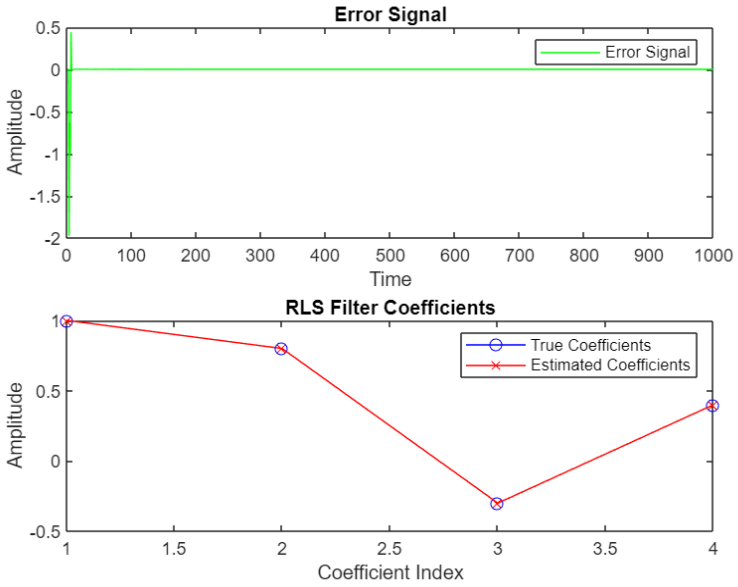

Figure 2 *: Signal de sortie et désiré avec le signal d'erreur superposé*

On observe que le signal d'erreur tend significativement vers zéro à partir d'un certain moment. Cela indique que l'algorithme a correctement ajusté les coefficients du filtre, de manière à ce que l'erreur

3. Test de l'algorithme RLS avec un signal simul´

Le but de cette partie est de comparer l'influence de l'ordre du filtre P et le pas mu sur le signal d'erreur. On a alors pris **P = 5, 10**. et **λ = 0.1, 0. 5, 0.9.**

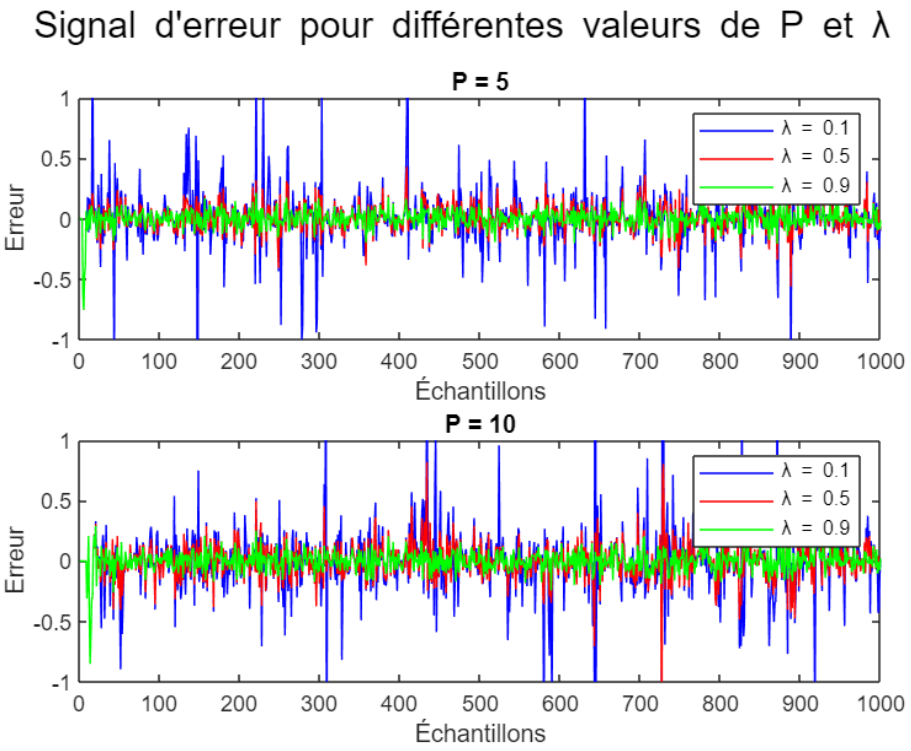

Figure 3** : Signal d'erreur pour **P = 5, 10**. et **λ = 0.1, 0. 5, 0.9.

Longueur du filtre P : On observe une réduction globale de l'amplitude des fluctuations de l'erreur, notamment pour les valeurs de λ=0.5 et λ=0.9. donc une valeur élevée de P (P = 10) réduit les fluctuations de l'erreur en permettant une meilleure adaptation au bruit, mais peut ralentir la convergence.

Facteur d'oubli λ : Un λ élevé (λ = 0.9) stabilise l'erreur en rendant l'algorithme moins sensible aux variations instantanées, bien que cela diminue la capacité d'adaptation rapide.

4. Bilan et comparaison des performances du LMS et RLS.

Critère	Algorithme LMS	Algorithme RLS
Vitesse de convergence	Dépend de μ: lente pour μ faible, rapide mais instable pour μ élevé.	Rapide, même pour des valeurs élevées de P.
longueur du filtre P	La convergence ralentit avec un P plus élevé	Bonne performance même avec un P élevé

Impact du pas μ	Faible μ (ex : 0.01) : convergence stable mais lente. Élevé μ (ex : 0.5) : convergence rapide mais risque d'instabilité.	Pas d'impact direct, car l'algorithme n'utilise pas de paramètre μ ; ajustement automatique des coefficients pour minimiser l'erreur.

Partie 2 : Soustraction Adaptative de Bruit (ANC)

1. Modèle de données

Le modèle de données considère un signal bruité défini comme :

$$d(n) = s(n) + N_0(n) \tag{1}$$

où :

- $s(n)$ est le signal propre (sans bruit),
- $N_0(n)$ représente un bruit ajouté au signal.

Pour estimer le bruit et le soustraire du signal, on utilise un signal de référence $x(n)$ défini comme :

$$x(n) = N_1(n) \tag{2}$$

Le signal de référence $x(n)$ est un bruit $N_1(n)$ qui est non corrélé avec le signal propre $s(n)$. Ce signal de référence est utilisé dans l'algorithme RLS pour estimer le bruit $N_0(n)$ et ainsi améliorer la qualité du signal $s(n)$.

2. Mise en œuvre

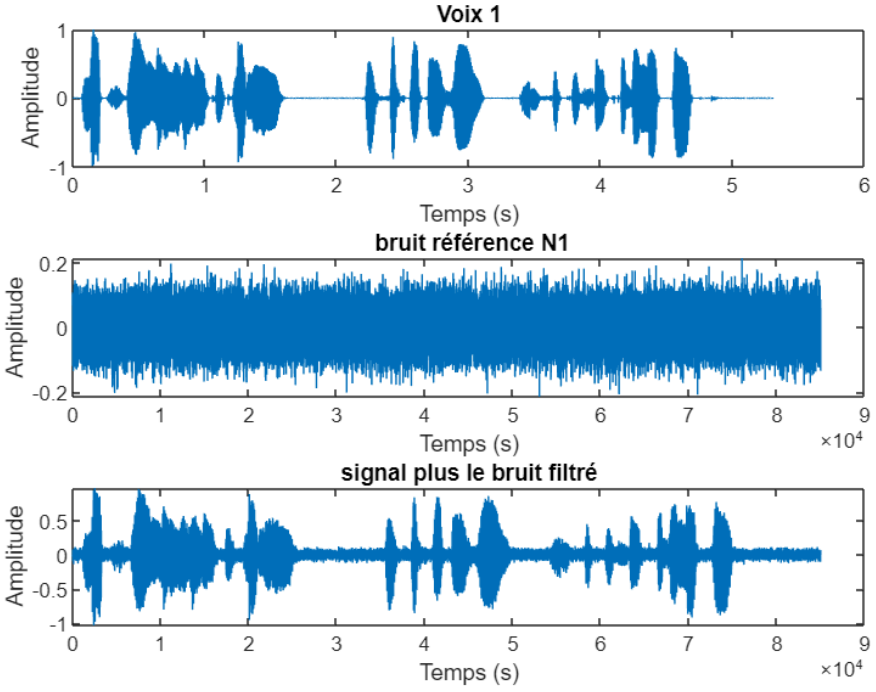

Figure 4 : *Voix1 , bruit référence N1 et Voix1 bruité*

Algorithm 1 Soustraction Adaptative de Bruit avec RLS

1: **Charger le signal** depuis `signal.dat` et le tracer.
2: **Générer le bruit de référence** N_1 comme un bruit blanc et le tracer.
3: **Filtrer le bruit** : Utiliser un filtre FIR d'ordre 32 et de fréquence 0.5 pour obtenir N_0 à partir de N_1.
4: **Ajouter le bruit au signal** : Obtenir le signal bruité $d(n) = s(n) + N_0$ et le tracer.
5: **Estimer le signal propre** $s(n)$ en appliquant l'algorithme RLS, puis tracer le signal estimé et l'erreur.

En appliquant l'algorithme precedant on obtient les traces suivantes, et on remarque le signal Voix1 a été bien estimé par l'algorithme RLS et que l'erreur tend vers 0 .

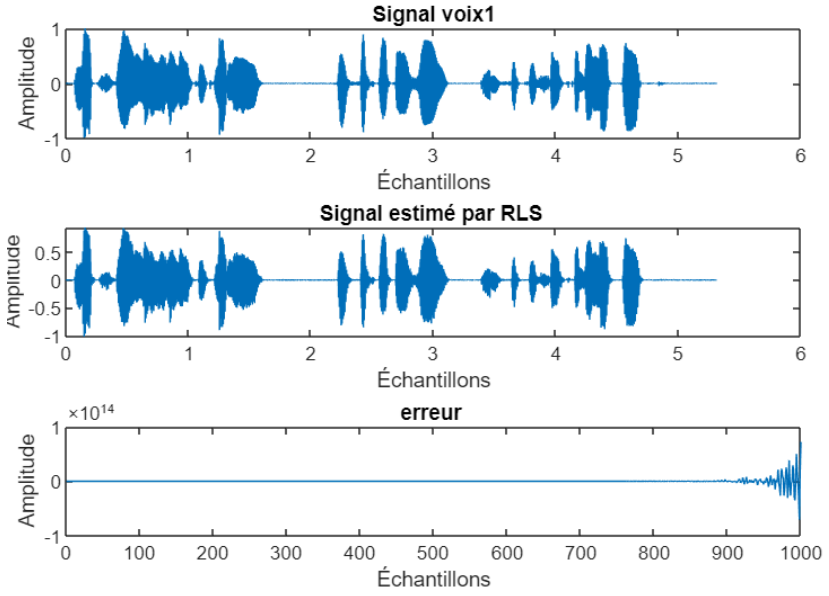

Figure 4 *: Voix1 ,Voix1 estimé par l'algo RLS et l'erreur*

Partie 3 : Algorithme NLMS

1 Équations de l'algorithme NLMS

1.1 Sortie du filtre

La sortie du filtre à l'itération n est donnée par :

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) \tag{1}$$

où $\mathbf{w}(n)$ représente le vecteur des coefficients du filtre et $\mathbf{x}(n)$ est le vecteur des signaux d'entrée.

1.2 Erreur

L'erreur à l'itération n est définie comme :

$$e(n) = d(n) - y(n) \tag{2}$$

où $d(n)$ est le signal désiré.

1.3 Mise à jour des coefficients

Les coefficients du filtre sont mis à jour selon la formule :

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{e(n)}{\epsilon + \|\mathbf{x}(n)\|^2} \mathbf{x}(n) \tag{3}$$

où μ est le pas de convergence, ϵ est une petite constante pour éviter la division par zéro, et $\|\mathbf{x}(n)\|^2$ est l'énergie du signal d'entrée.

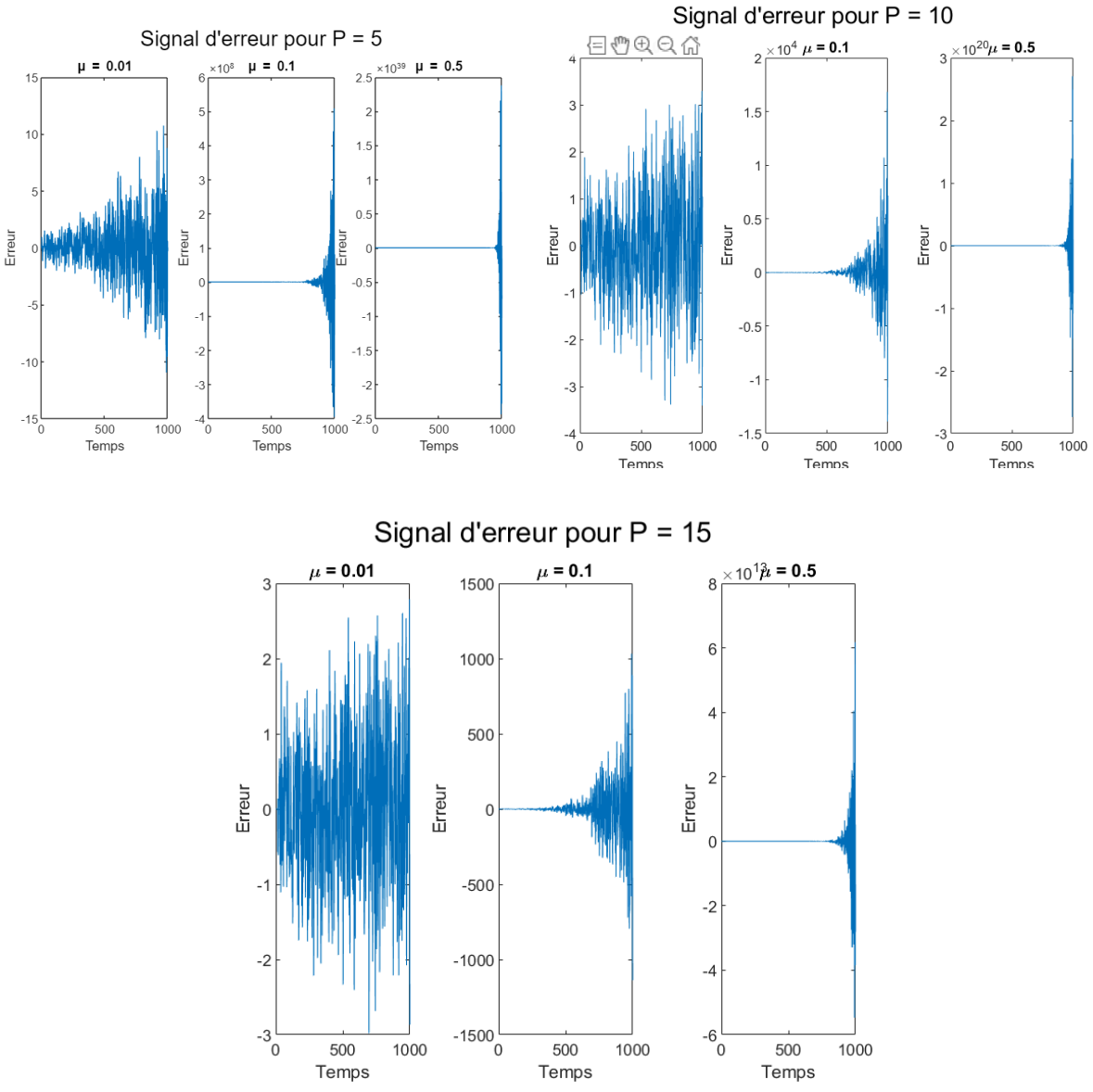

Figure 5** : Signal d'erreur pour **P = 5, 10, 15**. et **λ = 0.1, 0. 5, 0.9.

2. Bilan et comparaison des performances du NLMS, LMS et RLS.

Critère	Algorithme LMS	Algorithme RLS	Algorithme NLMS
Vitesse de convergence	Dépend de μ: lente pour μ faible, rapide mais instable pour μ élevé.	Rapide, même pour des valeurs élevées de P.	Convergence plus rapide que LMS, car le pas est normalisé, réduisant le risque d'instabilité.
longueur du filtre P	La convergence ralentit avec un P plus élevé	Bonne performance même avec un P élevé	Moins sensible à P par rapport à LMS, car la normalisation du pas aide à maintenir la stabilité.
Impact du pas μ	Faible μ (ex : 0.01) : convergence stable mais lente. Élevé μ (ex : 0.5) : convergence rapide mais risque d'instabilité.	Pas d'impact direct, car l'algorithme n'utilise pas de paramètre μ ; ajustement automatique des coefficients pour minimiser l'erreur.	La normalisation ajuste automatiquement μ en fonction de l'énergie du signal d'entrée, offrant une convergence plus stable