TP3 de Méthodes de Signal Avancées
Soustraction Adaptative de Bruit

I. Fijalkow et A.M. Masucci

Compte-rendu attendu à la fin des 4h.

- *Mise en équation de l'algorithme RLS.*
- *Listing du programme.*
- *Résultats de simulations commentés soit à la main sur les figures, soit en pdf.*

I. Partie 1

1) Rappeler les equations de l'algorithme RLS.

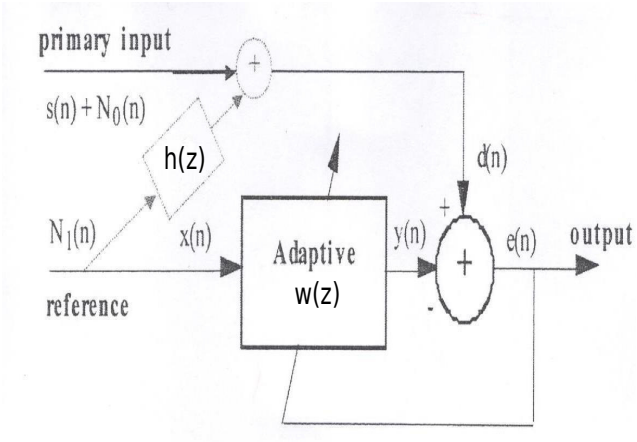
2) Mise en œuvre de l'algorithme RLS à partir de signaux x et d pour l'adaptation d'un filtre w de RIF à l'aide du logiciel Matlab:
   - Initialization;
   - Mise à jour;
   - Rapresentation des sorties.

3) Validation de l'algorithme RLS.
   Le signal x est tout d'abord un bruit blanc et le signal représentant d est obtenu par filtrage de x par le filtre de réponse impulsionnelle finie $\mathbf{h} = [1\ 0.8\ -0.3\ 0.4]^t$ supposé inconnu. Utiliser l'algorithme RLS pour calculer le filtre et le signal de sortie.
   - Tracer l'erreur e_n en fonction du temps.
   - Tracer les vrais coefficients et les estimations obtenues par le RLS.

4) Test de l'algorithme RLS avec un signal simulé.
   Le signal x est un bruit blanc et le signal représentant d est obtenu par filtrage de x plus un bruit. Le filtrage de x est obtenu par le filtre de réponse impulsionnelle finie de ordre P et fréquence 0.5. Utiliser l'algorithme RLS pour calculer le signal de sortie.
   - Étudier l'effect de la longueur P du filtre sur les performances obtenu, $P = 5, 10$.
   - Puor P fixé étudier l'effect du choix de le paramètre d'oubli $\lambda = 0.1, 0.5, 0.9$.

5) Bilan et comparaison des performances du LMS et RLS.

II. PARTIE 2



On suppose le modèle de données suivant: $d(n)$ qui est la somme d'un signal $s(n)$ et d'un bruit $N_0(n)$, et $x(n)$ qui est un signal de réference $N_1(n)$.
La Soustraction Adaptative de Bruit (Adaptive Noise Canceller (ANC) en anglais) a deux signaux d'entrée: un signal $s(n)$ bruité par un bruit $N_0(n)$ et un signal de réference $N_1(n)$. Le bruit $N_0(n)$ n'est pas corrélé avec le signal $s(n)$ mais il est obtenu par le filtrage du signal $N_1(n)$ par un filtre de réponse impulsionnelle finie h. En utilisant un filtrage adaptatif du signal $N_1(n)$, il est possible d'estimer le signal $s(n)$.

Mise en œuvre de l'application de soustraction adaptative de bruit:

- Charger le signal dans le file *signal.dat*.
- Tracer le signal.
- Créer le bruit référence N_1 comme un bruit blanc.
- Tracer le bruit référence.
- Créer le bruit N_0 corrélé au bruit N_1:
  N_0 est donnée par le filtrage de N_1 par un filtre de réponse impulsionelle finie de ordre 32 et fréquence 0.5.
- Ajouter le bruit N_0 au signal.
- Tracer le signal plus le bruit filtré.
- Estimer le signal en utilisant l'algorithme RLS.
- Tracer le signal original et le signal estimé par RLS.

III. PARTIE 3

1) Dérivation des équations de l'algorithme NLMS à partir de signaux x et d.
2) Mise mise en œuvre de l'algorithme NLMS à partir de signaux x et d pour l'adaptation d'un filtre w de RIF à l'aide du logiciel Matlab:
   - Initialization;
   - Mise à jour;

- Rapresentation des sorties.

3) Test de l'algorithme NLMS avec un signal simulé.
   Le signal x est un bruit blanc et le signal représentant d est obtenu par filtrage de x plus un bruit. Le filtrage de x est obtenu par le filtre de réponse impulsionnelle finie de ordre P et fréquence 0.5. Utiliser l'algorithme NLMS pour calculer le signal de sortie.
   - Étudier l'effect de la longueur P du filtre sur les performances obtenu, $P = 5, 10, 20$.
   - Puor P fixé étudier l'effect du choix de μ sur la vitesse de convergence.
4) Bilan et comparaison des performances du NLMS, LMS et RLS.

IV. Aide à la Programmation

Les commandes Matlab par réaliser le filtrage :

- Les filtres sont représentés dans le domaine complexe de z par la fonction de transfert suivante:

$$H(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}}{a(1) + a(2)z^{-1} + \cdots + a(m+1)z^{-m}},$$

  donc pour définir un filtre, il faut connaître les coefficients du numérateur et du dénominateur.
  Sous MATLAB, ces coefficients doivent être rangés dans deux vecteurs lignes a et b, puis, pour effectuer le filtrage, il suffit de taper :
  y = **filter**(b,a,X) filtre les données dans le vecteur *X* par le filtre obtenu par le vecteur *b* (coefficients du numérateur) et le vecteur *a* (coefficients du dénominateur).
- La fonction **fir1**$(P-1, Wn)$ permet de réaliser un filtre de réponse impulsionnelle finie (FIR) d'ordre p. Elle génère les coefficients b seulement (donc $a = 1$). Il suffit de choisir l'ordre P du filtre et les fréquences rangées dans un vecteur W_n dans l'ordre croissant.