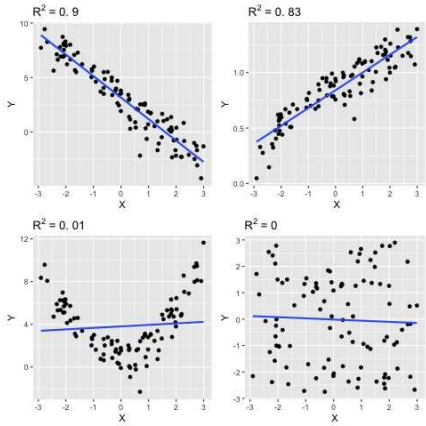
Chapitre III : Analyse spectrale paramétrique et filtrage numérique adaptatif

Introduction

La modélisation paramétrique consiste à associer à un signal un modèle, représenté par un vecteur dit paramètre $\theta = [\theta_1, \theta_2, ..., \theta_p]$ censé représenter au mieux le signal considéré. Elle nécessite le choix d'un modèle qui ne peut se faire que si l'on possède des informations sur le signal.



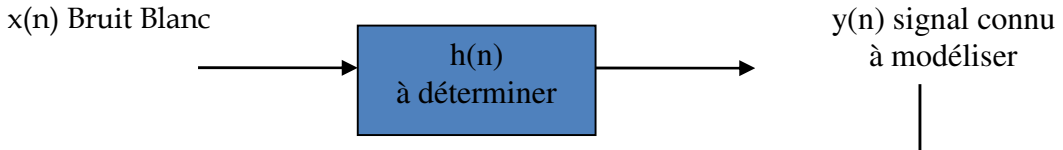
Elle permet entre autres : de représenter les données par un vecteur de plus petite dimension. Ce qui pourra servir lors de la compression et transmission, comme signature pour une identification ou classification.

Nous aborderons dans ce cours le cas de la modélisation spectrale qui nous permettra d'estimer le spectre du signal.

1. Modélisation spectrale paramétrique

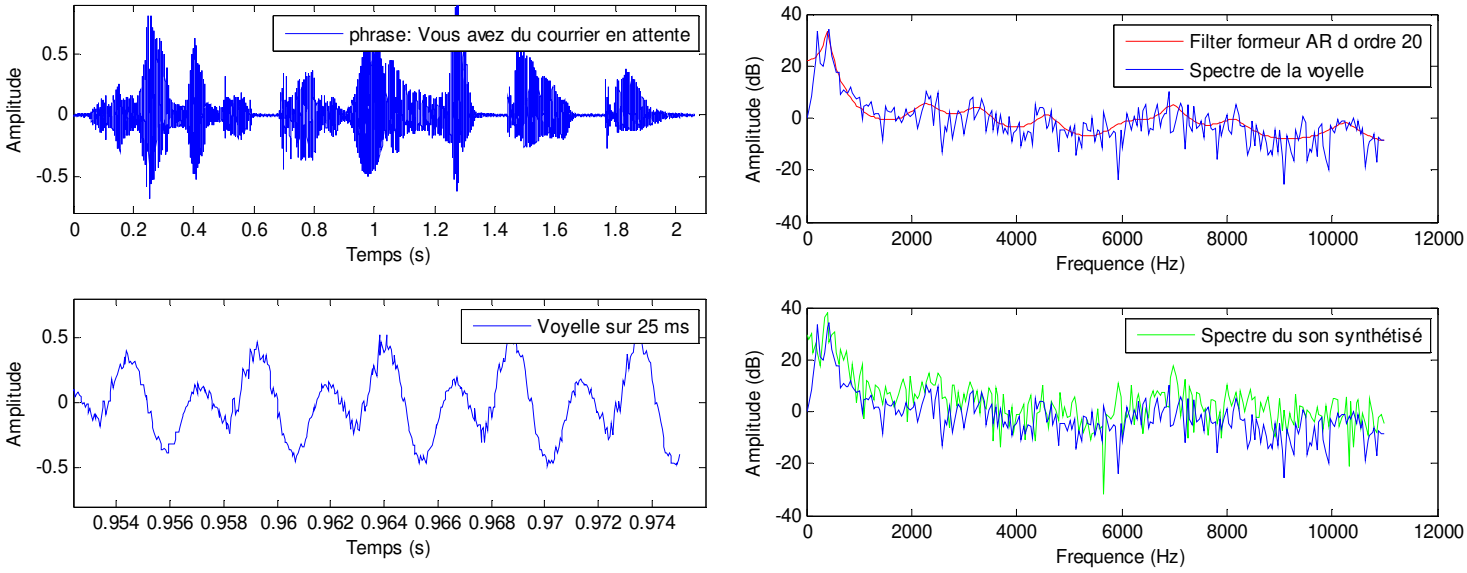
Nous avons vu précédemment qu'un signal aléatoire peut être modélisé (synthétisé) comme la réponse d'un filtre linéaire à une excitation sous forme de bruit blanc tel que $|H(f)|^2 = S_x(f)/\sigma_b{}^2$. Ce filtre formeur H(f) est aussi dit processus générateur. Ces paramètres associés à la variance du bruit σ_b^2 constituent le modèle mathématique correspondant au signal aléatoire. Le concept de processus générateur de signal a été particulièrement développé et appliqué avec des filtres numériques. Ce sont les modèles de signaux les plus utilisés en traitement statistique du signal (estimation, prédiction...). Selon la nature du filtre, on peut obtenir différents modèles de signaux (AR,MA, ARMA, etc.)

Exemple : Lors d'un appel par GSM (téléphone portable), le portable qui fait office d'un micro- ordinateur regroupant différentes fonctionnalités dont l'analyse, la synthèse, le codage, etc. va nous permettre de modéliser la parole (aléatoire) en opérant un codage LPC par tranches de dizaines de ms. Elle consiste à retrouver les paramètres du filtre formeur *h(n)* pour chaque tranche *y(n)* enregistrée et analysée. Ce sont ces paramètres (a_i et b_j) qui seront transmis pour produire un signal de synthèse approchant le signal original y(n).



$$H(z) = \frac{\sum_{j=0}^{M} b_j z^{-j}}{\sum_{i=0}^{M} a_i z^{-i}} \qquad S_y(f) = \sigma_b{}^2 |H(f)|^2$$

Ci-dessous la phrase "vous avez du courrier en attente" échantillonnée à une fréquence fe =22050 (45531 échantillons) et un zoom sur une voyelle de durée 25 ms (500 échantillons).



En comparant l'allure du filtre formeur H(f) à celle de Sy(f), on note que l'on retrouve l'allure générale du spectre de la voyelle, notamment les fréquences dont la puissance est maximale. Sachant que le spectre de la voyelle comporte 500 valeurs et que le filtre H(f) équivalent est obtenu à partir de 20 coefficients, il vaut mieux transmettre 21 coefficients (20 a_i + variance du bruit) que 500 valeurs.

C'est ainsi que les modèles autorégressifs sont d'un emploi de plus en plus répandu en traitement du signal : codage et transmission par prédiction linéaire, synthèse de parole, reconnaissance, etc.

Remarque : Le signal de parole est un processus aléatoire non-stationnaire à long terme, mais il est considéré comme stationnaire dans des fenêtres temporelles d'analyse de l'ordre de 20 à 30ms. Cette propriété de stationnarité à court terme permet donc une analyse et modélisation progressive du signal de parole Pour éviter toutes pertes d'information, on veillera à prendre des fenêtres chevauchantes.

2. Modèle auto-régressif (AR)

Les signaux autorégressifs sont obtenus par passage d'un bruit blanc dans un filtre purement récursif. Ce filtre est donc de réponse impulsionnelle infinie.

$$H(z) = 1/\left(1 + \sum_{i=1}^{N} a_i z^{-i}\right)$$

A partir de H(z), on peut déterminer l'équation aux différences : $y(n) = x(n) - \sum_{i=1}^{N} a_i y(n-i)$

Cela signifie que le signal y(n) est supposé être prédictible en fonction d'un certain nombre de ses valeurs antérieures.

Sachant que x(n) est un bruit blanc alors d'où $R_{xx}(k) = \sigma^2 \delta(k)$

- $\mu_x = E\{x(n)\} = 0$ - $R_{xx}(0) = E\{x(n)^2\} = \sigma^2$ - $R_{xx}(k) = E\{x(n)x(n-k)\} = 0 \ \ pour \ \ k \neq 0$

Calculons alors $R_{yy}(k)$

$$R_{yy}(k) = E\{y(n)y(n-k)\} = E\{x(n)y(n-k)\} - \sum_{i=1}^{N} a_i E\{y(n-i)y(n-k)\}$$

$$= E\left\{x(n)\sum_{k'} x(n-k-k')h(k')\right\} - \sum_{i=1}^{N} a_i R_{yy}(k-i) = \sum_{k'} E\{x(n)x(n-k-k')\}h(k') - \sum_{i=1}^{N} a_i R_{yy}(k-i)$$

$$= \sum_{k'} R_{xx}(k+k')h(k') - \sum_{i=1}^{N} a_i R_{yy}(k-i) = R_{xx}(k)h(0) + R_{xx}(k+1)h(1) + ... - \sum_{i=1}^{N} a_i R_{yy}(k-i)$$

-Si k=0, $R_{yy}(0) = R_{xx}(0).h(0) - \sum_{i=1}^{N} a_i R_{yy}(-i) = \sigma^2.1 - \sum_{i=1}^{N} a_i R_{yy}(i)$

- Pour k=1 à N, $R_{yy}(k) = 0 - \sum_{i=1}^{N} a_i R_{yy}(k-i) = -\sum_{i=1}^{N} a_i R_{yy}(k-i)$

On peut utiliser une forme matricielle : R. a = S (Rappelons que pour un signal réel $R_{yy}(k)=R_{yy}(-k)$

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & ..... & R_{yy}(N) \\ R_{yy}(1) & R_{yy}(0) & ..... & R_{yy}(N-1) \\ & & ..... & \\ R_{yy}(N) & R_{yy}(N-1) & ..... & R_{yy}(0) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ a_1 \\ . \\ a_N \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ . \\ 0 \end{bmatrix}$$

R est la matrice d'autocorrélation dont le terme général r_{ij} ne dépend que de la différence *i-j* (Matrice de Toeplitz). La résolution de ces équations dites de Yule-Walker permet de connaître les paramètres du filtre et la variance du bruit blanc.

On peut reformuler cette matrice sous la forme suivante (On l'emploiera en TP) :

$$\overbrace{\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & ..... & R_{yy}(P-1) \\ R_{yy}(1) & R_{yy}(0) & ..... & R_{yy}(P-2) \\ & & ..... & \\ R_{yy}(P-1) & R_{yy}(P-2) & ..... & R_{yy}(0) \end{bmatrix}}^{C} \cdot \begin{bmatrix} a_1 \\ a_2 \\ . \\ a_P \end{bmatrix} = \overbrace{\begin{bmatrix} -R_{yy}(1) \\ -R_{yy}(2) \\ . \\ -R_{yy}(P) \end{bmatrix}}^{B}$$

$$\sigma^2 = R_{yy}(0) + \sum_{i=1}^{P} a_i R_{yy}(i)$$

Remarques : Nous ne disposons pas d'un processus aléatoire mais d'une seule réalisation soit *y(n),* il n'est pas donc pas possible de calculer l'auto-corrélation statistique $R_{yy}(k)$. Cette dernière sera remplacée par l'autocorrélation temporelle en faisant l'hypothèse que le processus est ergodique (voir exo 1 du TP n°4).

Il existe divers algorithmes (Burg, Levinson) qui permettent d'estimer assez rapidement les a_i et σ sans passer par l'inversion matricielle. Tout comme il est possible de déterminer l'ordre N adéquat (critère AIC).

Exemple d'application

1. On considère le modèle auto-régressif (AR) d'ordre 1 tel que : $x(n)=-a_1.x(n-1)+b(n)$

- Déterminer les équations de Yule-Walker pour ce modèle
- En supposant que x(n) est connu, déterminer les paramètres du modèle.
- Déterminer les $R_x(k)$ (les $a_i$sont supposés connus)

Réponses :

- ($a_1=-Rx(1)/Rx(0)$ $\sigma^2=Rx(0)(1-a_1^2)$)
- ($Rx(0)= \sigma^2/(1-a_1^2)$ $Rx(1)=-a_1\sigma^2/(1-a_1^2)Rx(k)=(-a_1)^k\sigma^2/(1-a_1^2)$)

2. Refaire le même travail pour un modèled'ordre 2 tel que : $x(n)=-a_1.x(n-1)-a_2.x(n-2)+b(n)$

Réponses :

- $a_1=Rx(1)[Rx(2)-Rx(0)]/[Rx(0)^2-Rx(1)^2]$ $a_2=[Rx(1)^2-Rx(0)Rx(2)]/[Rx(0)^2-Rx(1)^2]$
- $\sigma^2=Rx(0)+Rx(1)^2[Rx(2)-Rx(0)]/[Rx(0)^2-Rx(1)^2]+Rx(2)[Rx(1)^2-Rx(0)Rx(2)]/[Rx(0)^2-Rx(1)^2]$
- $Rx(1)=-a_1Rx(0)/(a_2+1)$ $Rx(2)=(-a_2+a_1^2/(1+a_2))Rx(0)Rx(0)=(1+a_2)\sigma^2/(1+a_2- a_1^2-a_2^3+a_2a_1^2)$

-Algorithme de Levinson

C'est un algorithme qui permet de résoudre tout système du type Ax = b avec A Toeplitz, donc en particulier les équations normales de Yule-Walker :

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & ..... & R_{yy}(N) \\ R_{yy}(1) & R_{yy}(0) & ..... & R_{yy}(N-1) \\ & & ..... & \\ R_{yy}(N) & R_{yy}(N-1) & ..... & R_{yy}(0) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ a_1 \\ . \\ a_N \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ . \\ 0 \end{bmatrix}$$

Le principe d l'algorithme de Levinson est de trouver de façon récursive les paramètres d'ordre k en fonction des paramètres d'ordre k − 1.

- Initialisation : $a_1(1) = k(1) = -\frac{R_{yy}(1)}{R_{yy}(0)}$

$$\sigma_1^2 = (1 - |a_1|^2)R_{yy}(0)$$

- Récursion : pour *k* allant de 2 à P (ordre du modèle AR)

$$a_k(k) = -\frac{R_{yy}(k)+\sum_{i=1}^{k-1} a_{k-1}(i)R_{yy}(k-i)}{\sigma_{k-1}^2} = k(k)$$

$$a_k(i) = a_{k-1}(i) + a_k(k)a_{k-1}(k-i) \quad pour\ i = 1, \dots\dots\dots, k-1$$

$$\sigma_k^2 = (1 - |a_k(k)|^2)\sigma_{k-1}^2$$

3. Modèle à moyenne ajustée (MA)

Les signaux à moyenne mobile sont obtenus par passage d'un bruit blanc dans un filtre purement transverse.

Ce filtre est aussi appelé filtre à réponse impulsionnelle finie : $H(z)=\sum_{i=0}^{M} b_i z^{-i}$

Le signal *y(n)* est supposé pouvoir s'écrire comme une combinaison linéaire d'échantillons décorrélés entre eux, ce qui peut se formaliser comme une combinaison linéaire d'échantillons d'un bruit blanc *x(n)*.

On a donc : $y(n)=\sum_{i=0}^{M} b_i x(n-i)$

et $\mu_y = E\{y(n)\} = \sum_{i=0}^{M} b_i \mu_x = \mu_x \sum_{i=0}^{M} b_i = \mu_x . H_f(0)$

On cherche les paramètres du filtre qui génèrent y(t) à partir de x(t), bruit blanc centré :

$$R_{yy}(k) = E\{y(n)\,y(n-k)\} = E\left\{\sum_{i=0}^{M} b_i x(n-i).\sum_{j=0}^{M} b_j x(n-j-k)\right\}.$$

$$R_{yy}(k) = \sum_{i=0}^{M} b_i . \sum_{j=0}^{M} b_j E\{x(n-i).x(n-j-k)\} = \sum_{i=0}^{M} b_i . \sum_{j=0}^{M} b_j R_{xx}(j+k-i)$$

- Si j+k≠i $\Rightarrow R_{yy}(k)=0$
- Sinon $\Rightarrow R_{yy}(k)=\sigma^2 \sum_{j=0}^{M-k} b_{j+k}.b_j$

Le problème est non linéaire en fonction des coefficients, il faut un algorithme de programmation non linéaire pour obtenir b_i à partir des $R_{yy}(k)$. Cependant, l'algorithme de Durbin permet d'approcher la solution optimale avec de bons résultats. Le principe de cet algorithme consiste à identifier le modèle MA d'ordre M avec un modèle AR d'ordre N>>M (TP n°4). En effet, tout modèle MA peut être identifié à un modèle AR d'ordre infini: $\sum_{i=0}^{M} b_i z^{-i} = 1/\sum_{i=0}^{\infty} a_i z^{-i}$

Exemple

On considère le modèle à moyenne ajustée (MA) :

A. d'ordre 1 tel que $x(n)=e(n)+b_1 . e(n-1)$

- Calculer μ_x.

- En supposant que x(n) est connu, déterminer les paramètres du modèle.
- Connaissant les paramètres du modèle, déterminer les $R_x(k)$

B. d'ordre 2 tel que : $x(n)=e(n)+b_1.\ e(n-1)+b_2.\ e(n-2)$

- Calculer μ_x.
- Connaissant les paramètres du modèle, déterminer les $R_x(k)$ Réponses

$\mu_x=0$ $Rx(0)=(1+b_1^2).\sigma^2$ $Rx(1)=b_1.\sigma^2$ $Rx(k)=0$ pour $k\geq 2$

$b_1=(Rx(0)\pm sqrt(Rx(0)^2-4Rx(1)^2)/2Rx(1)$ $\sigma^2=2/(Rx(0)\pm sqrt(Rx(0)^2-4Rx(1)^2)$

$Rx(0)=(1+b_1^2+b_2^2).\sigma^2$ $Rx(1)=(b_1+b_1\,b_2).\sigma^2$ $Rx(2)=b_2.\sigma^2 Rx(k)=0$ pour $k\geq 3$

Remarque : Il est très important de remarquer que nous ne disposons que d'une seule réalisation du signal aléatoire à modéliser *y(n)*, de ce fait l'auto-corrélation statistique $R_{yy}(k)$ est obtenu de l'auto-corrélation temporelle en considérant le processus ergodique.

4. Modèle ARMA

Les signaux ARMA sont obtenus par passage d'un bruit blanc dans un filtre récursif appelé aussi filtre à réponse impulsionnelle infinie (R.I.I). Ces signaux sont une combinaison des signaux AR etMA. La fonction de transfert du filtre présente un numérateur et un dénominateur:

Soit $y(n)=\sum_{i=0}^{M} b_i x(n-i)-\sum_{i=1}^{N} a_i y(n-i)$ $\qquad H(z)=\sum_{i=0}^{M} b_i z^{-i} \Big/ \left(1+\sum_{i=1}^{N} a_i z^{-i}\right)$

La corrélation statistique de y(n) s'écrit alors : $R_{yy}(k)=-\sum_{i=1}^{N} a_i R_{yy}(k-i)+\sigma^2 \sum_{j=0}^{M-k} b_{j+k}.b_j$

C'est une équation non linéaire en a_i et b_j.

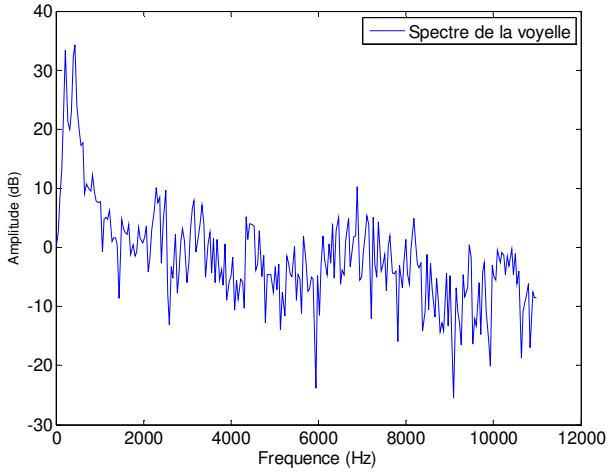
La modélisation ARMA peut se décomposer en une modélisation AR suivie d'une modélisation MA. Le modèle AR présente une simplicité de calcul par rapport aux modèles MA et ARMA du fait où les coefficients AR sont solutions d'un système linéaire d'équations. Alors que la détermination des coefficients MA et ARMA requiert la résolution d'équations non linéaires. Cependant, le modèle ARMA permet de modéliser aussi bien les minima que les maxima de la DSP et est donc moins restrictif que le modèle AR.

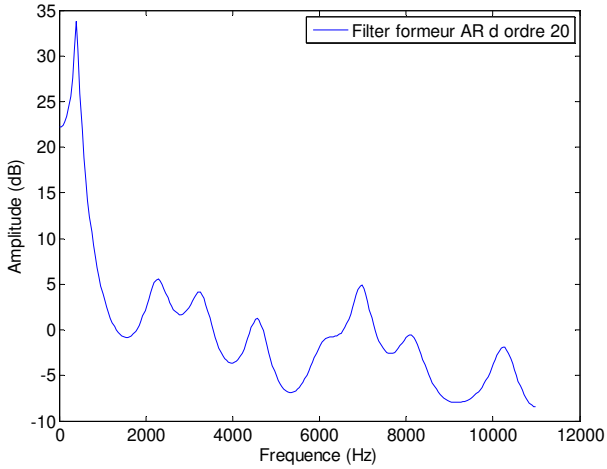
Les applications des modèles AR, MA, ARMA sont nombreuses, entre autres :

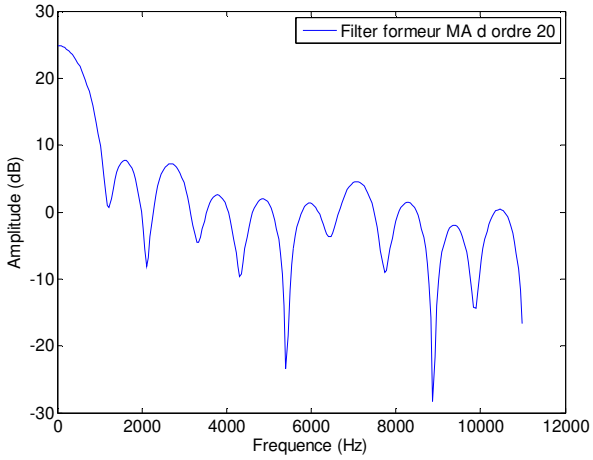
- la modélisation et la prédiction de série temporelle dite séries chronologiques Une série chronologique est une suite formée d'observations au cours du temps que l'on cherche à modéliser pour la prédiction de données futures. Ainsi, en finance, cela permet de modéliser le cours des devises ou du pétrole. Alors qu'en météorologie, les cela permet de faire des prévisions sur la température ou les précipitations. Dans chacun des cas, on essaiera à partir d'un échantillon de données de construire le meilleur modèle qui s'ajuste ces données.

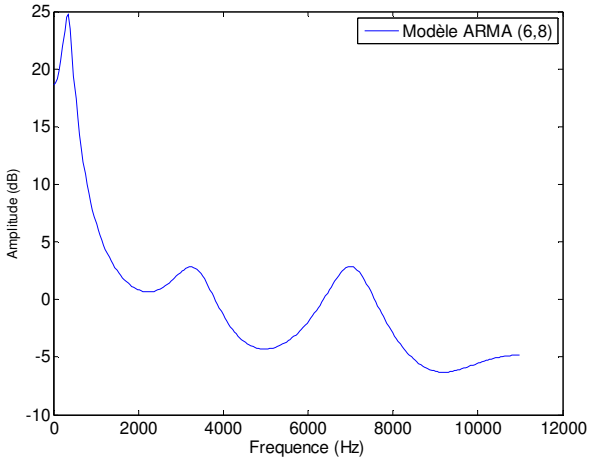
- l'estimation du spectre d'un signal aléatoire, etc. Cette dernière application est basée sur l'identification des paramètres du modèle considéré : Le modèle AR est bien adapté aux signaux composés de raies pures dans du bruit blanc. Alors que le modèle MA est bien adapté aux signaux dont la puissance est nulle dans certaines bandes de fréquences.

Exemple : Reprenons à nouveau l'exemple de parole, pour lequel, nous avons testé les trois variantes









Rappelons que pour un ordre du filtre faible, le modèle ne sera pas en mesure de capturer toutes les informations pertinentes du signal, à contrario s'il est trop élevé, il modélisera le bruit. Rappelons qu'avec le périodogramme (Algorithmes rapides : FFT) et ses variantes :

- Résolution en *fe/N*=> Difficile de séparer 2 fréquences proches,

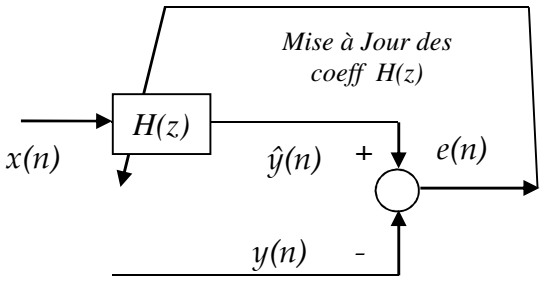
- Lobes secondaires occultant les fréquences à faibles amplitudes

Nous n'irons pas plus loin, dans le cadre de ce cours, le but n'étant que d'introduire les notions de modélisation et de prédiction linéaire.

6. Filtrage Numérique adaptatif

Dans le filtrage de Wiener, le critère d'optimalité est stochastique : on désire minimiser la moyenne de l'erreur au carré. Cela requiert des statistiques de second ordre des processus (moyennes et corrélations). En pratique, la matrice de corrélation $R_{xx}(k)$ et l'inter-corrélation $R_{yx}(k)$ ne sont pas connues et doivent donc être estimés à partir des données observées (temporelles) en supposant les processus ergodiques.

Le second problème est lié à l'hypothèse même de stationnarité qui n'est pas vérifiée à long terme. Le filtrage de Wiener devient inadéquat pour les situations dans lesquelles le signal ou le bruit sont non stationnaires. Dans de telles situations le filtre optimal doit être variable dans le temps. La solution à ce problème est fournie par le filtrage adaptatif.



Le filtrage adaptatif comporte une mise à jour récursive des paramètres (coefficients) du filtre. L'algorithme part de conditions initiales prédéterminées et modifie de façon récursive les coefficients du filtre pour s'adapter au processus. Pour cela, les coefficients de la réponse impulsionnelle du filtre sont adaptés en fonction de l'erreur par une boucle de retour.

- L'algorithme du gradient fournit un algorithme récursif de calcul des coefficients du filtre de Wiener (sans passer par l'inversion de R_{xx}).
- L'algorithme du gradient stochastique (LMS) est une version déterministe dans laquelle les grandeurs statistiques impliquées sont remplacées par des valeurs instantanées.
- L'algorithme des moindres carrés (RLS) qui est un algorithme récursif du LMS basé sur la minimisation de la somme des erreurs quadratiques sur un laps de temps donné.

6.1. Algorithme du Gradient (de descente)

On cherche à calculer les paramètres du filtre linéaire optimal.

- A l'instant n, ces paramètres sont notés $h(n)=\left[b_0(n) \quad b_1(n) \quad \ldots \quad b_{N-1}(n)\right]^T$ et
- les entrées du filtre forment le vecteur $\mathrm{X}(n) = [x(n), x(n-1), \ldots \ldots x(n-N+1)]^T$.
- A chaque instant n, la sortie du filtre est un signal $\hat{y}(n)$ que l'on souhaite aussi proche que possible du signal original y(n).

On rappelle que la solution optimale donnée par la solution des équations de Wiener-Hopf est : $R_{xx}h = R_{yx}$

avec $\xi(n) = E\{e^2(n)\} = E\left\{\left(y(n) - \sum_{i=0}^{N-1} b_i x(n-i) \right)^2\right\}$

L'algorithme procède en trois étapes, une étape d'initialisation et deux étapes qui sont itérées :

1. Initialisation : au temps n = 0, on attribue une valeur initiale h(0)=[0, 0, ….. 0]

2. A chaque instant n (l'instant courant), on calcule le gradient de l'erreur (la dérivée) $\nabla_{h(n)}\xi(n)$ par rapport au vecteur h(n) : $\nabla_{h(n)}\xi(n) = -2R_{yx} + 2R_{xx}h(n)$

3. On ajuste les paramètres du filtre selon : $h(n+1) = h(n) - \frac{\mu}{2}\nabla_{h(n)}\xi(n) = h(n) + \mu\,(R_{yx} - R_{xx}h(n))$

$\frac{\mu}{2}$: un petit scalaire positif qui permet d'ajuster la vitesse et la précision des ajustements. A noter qu'il existe des versions à pas d'adaptation μ_n décroissant.

L'idée de cet algorithme est que les ajustements en sens opposé au gradient (descente du gradient) vont conduire h(n) vers la valeur h_{opt} qui est associée à l'erreur minimale ξ_{min}. Si la dérivée est positive alors on va diminuer $h(n)$ de manière à se déplacer vers le minimum de la fonction de coût. Le terme correctif sera donc choisi négatif et vice-versa

L'algorithme est stable si $0 < \mu < \frac{2}{\lambda_{max}}$ où λ_{max} est le max des valeurs propres de R_{xx}

Cet algorithme est aussi appelé algorithme du gradient déterministe puisque les quantités R_{yx} et R_{xx} sont parfaitement connues et déterminées. On parle aussi de filtrage de Wiener adaptatif.

6.1. Algorithme du Gradient stochastique (LMS)

La méthode de descente (gradient) est un algorithme qui estime, par itérations successives, une solution qui tend vers la solution optimale de l'équation de Wiener-Hopf, sans inverser la matrice R_{xx}. Néanmoins, à chaque itération, il faut calculer les corrélations statistiques R_{xx} et R_{yx} que nous avons remplacé par des corrélations temporelles sous hypothèse d'ergodicité.

L'algorithme adaptatif connu sous le nom de Least-Mean-Square (LMS) repose sur la simplification du gradient de $\xi(n)$ soit $\nabla_{h(n)}\xi(n) = -2R_{yx} + 2R_{xx}h$ en remplaçant les moyennes statistiques (ergodiques : temporelles) R_{xx} et R_{yx} par des estimations instantanées :

$$\hat{R}_{xx} = \mathrm{X}(n)\mathrm{X}^T(n)$$

$$\hat{R}_{yx} = y(n)\mathrm{X}(n)$$

Ainsi le gradient devient $\widehat{\nabla}_{h(n)}\xi(n) = -2y(n)\mathrm{X}(n) + 2\mathrm{X}(n)\mathrm{X}^T(n)h(n)$

A noter que minimiser l'erreur quadratique instantanée est moins efficace et moins précis que corriger l'erreur quadratique moyenne.

Ainsi, $\hat{h}(n+1) = \hat{h}(n) + \mu\, \mathrm{X}(n)[\overbrace{y(n) - \mathrm{X}^T(n)\hat{h}(n)}^{e(n)}]$

Soit $\hat{h}(n+1) = \hat{h}(n) + \mu\, \mathrm{X}(n)e(n)$

1. Initialisation : au temps n = 0, $\hat{h}(0) = [0, 0, \dots \dots \dots 0]$

2. Estimation du signal d'erreur : $e(n) = y(n) - \mathrm{X}^T(n)\hat{h}(n)$

3. Actualisation des coefficients : $\hat{h}(n+1) = \hat{h}(n) + \mu\, \mathrm{X}(n)e(n)$

L'algorithme LMS converge en moyenne, c'est-à-dire que, pour n→∞ , $\hat{h}(n) \rightarrow h_{opt}$ des équations de Wiener-Hopf. A noter que l'algorithme de descente du gradient tend vers l'erreur minimale ξ_{min} lorsque n tend vers l'infini et ceci de façon exponentielle régulière, en raison de l'estimation exacte des R_{xx} et R_{yx}. Au contraire, l'algorithme LMS, qui repose sur des approximations très grossières de ses quantités converge avec des oscillations chaotiques autour ξ_{min} , et ce en dépit d'un choix correct de μ.

6.3. Algorithme des moindres carrés

Les équations de Wiener-Hopf sont obtenues en minimisant l'erreur quadratique moyenne $\xi(\mathrm{n}) = \mathrm{E}\{e^2(\mathrm{n})\} = \mathrm{E}\{(\mathrm{y}(\mathrm{n}) - \hat{y}(\mathrm{n}))^2\}$. L'algorithme des moindres carrées est basé sur la minimisation de la somme des erreurs quadratiques : $\xi(n) = \sum_{k=1}^{n} \lambda^{n-k} e^2(k)$ sur un laps de temps donné.

Si = 1 , $\xi(n)$ est somme des erreurs quadratiques; Si $\lambda < 1$, les erreurs passées sont pondérées avec un facteur (d'oubli) qui décroît exponentiellement.

Version récursive du RMS,

- $\xi(n) = \sum_{k=1}^{n} \lambda^{n-k} e^2(n)$ où $\mathrm{e(k)} = \mathrm{y(k)} - \hat{y}(\mathrm{k}) = \mathrm{y(k)} - h^T(n)\mathrm{X}(k)$

$$R_{xx\lambda}(n)h(n) = R_{yx\lambda}(n)$$

Avec $R_{xx\lambda}(n) = \sum_{k=1}^{n} \lambda^{n-k} \mathrm{X}(k)\mathrm{X}^T(k) = \lambda \sum_{k=1}^{n-1} \lambda^{n-k-1} \mathrm{X}(k)\mathrm{X}^T(k) = \lambda R_{xx\lambda}(n-1) + \mathrm{X}(k)\mathrm{X}^T(k)$

et $R_{yx\lambda}(n) = \sum_{k=1}^{n} \lambda^{n-k} y(k)\mathrm{X}(k) = \lambda \sum_{k=1}^{n-1} \lambda^{n-k-1} y(k)\mathrm{X}(k) + y(k)\mathrm{X}(k) = \lambda R_{yx\lambda}(n) + y(k)\mathrm{X}(k)$

Pour $\lambda = 1$, la matrice $R_{xx\lambda}(n)$ est une approximation de la matrice d'auto-corrélation. Elle est symétrique telle que : $R_{xx\lambda}(n)_{ij} = \frac{1}{n}\sum_{k=1}^{n} x(k-i+1)\, x(k-j+1)$

L'algorithme se présente comme suit :

- Initialisation $R_{xx\lambda}^{-1}(0) = \left(\sum_{k=-n_0}^{0} \mathrm{X}(k)\mathrm{X}^T(k)\right)^{-1}$ et $R_{yx\lambda}(0) = \sum_{k=-n_0}^{0} y(k)\mathrm{X}(k)$

- $\hat{h}(n) = R_{xx\lambda}^{-1}(n)R_{yx\lambda}(n)$

avec $R_{xx\lambda}(n) = \lambda R_{xx\lambda}(n-1) + \mathrm{X}(k)\mathrm{X}^T(k)$ et $R_{yx\lambda}(\mathrm{n}) = \lambda R_{yx\lambda}(n) + y(k)\mathrm{X}(k)$

Remarque : Le filtrage des moindres carrés se place en déterministe. Alors que le filtre de Wiener utilise la méthode des moindres carrés en stochastique.

Exemples d'application

Le filtrage adaptatif (au sens de Wiener, au sens des moindres carrés, le filtrage de Kalman, le filtrage particulaire) est utilisé pour divers objectifs entre autres : la modélisation, la prédiction, l'identification des systèmes, l'annulation d'interférence (écho, bruit), etc. permettent la reconstitution d'un signal en milieu bruit

- L'annulation d'écho va nous fournir un premier exemple d'utilisation de l'algorithme LMS. Téléphone (micro et haut-parleur proches).
- Annulation d'un brouilleur tel un signal perturbé par un brouilleur sinusoïdal de fréquence connue mais dont on ignore l'amplitude A et la phase φ.

6.3. Autres filtres adaptatifs

- Kalman : Le filtre de Kalman est employé pour estimer des paramètres d'un système évoluant dans le temps à partir de mesures bruités. Il est particulièrement adapté pour le tracking du fait qu'il permet non seulement de prédire les paramètres mais aussi de rectification les erreurs de mesure et du modèle. Il opère de façon similaire aux filtres adaptatifs. La prédiction se fait en employant l'estimation précédente. Le calcul d'erreur permet, ensuite, de faire la mise à jour de cette prédiction grâce aux nouvelles mesures.

A noter que la version étendue (Filtre de Kalman étendu) permet de prendre en compte une modélisation non linéaire.

- Particulaire (méthode de Monté-Carlo séquentielle) : Une alternative aux filtres de Kalman étendu. E filtre a pour but d'estimer la densité postérieure des variables qu'on cherche à estimer (dites variables d'état) en fonction des observations (variables d'observation) en supposant que les variables d'état constituent une chaine de Markov de premier ordre.

Il est utilisé particulièrement pour des applications de navigation : suivi de cible, guidage de missile, robotique, etc.

TD n° 4 : Modélisation AR et Filtrage adaptatif

1. Soit un signal aléatoire y(n) SSL et ergodique dont l'autocorrélation temporelle $\overline{R}_y(k) = \alpha^{|k|}$ avec 0<α<1

- Identifier le modèle linéaire adéquat (AR ou MA) pour y(n).
- En supposant que le système h(n) est filtre purement récursif, donner le schéma du modèle en définissant l'entrée, le système, et la sortie.
- Rappeler les hypothèses nécessaires liées à l'emploi de ce modèle.
- On considère que le modèle est d'ordre 1, déterminer ses paramètres.

2. On considère le filtre linéaire à temps discret défini par $y(n) = x(n) + b_1 x(n-1) + b_2 x(n-2)$.
où X(n) et Y (n) désignent respectivement les processus aléatoires réels d'entrée et de sortie du filtre où b1 et b2 sont 2 coefficients réels. On suppose que x(n) est une suite de variables aléatoires centrées**,** indépendantes et de variance σ^2**.**

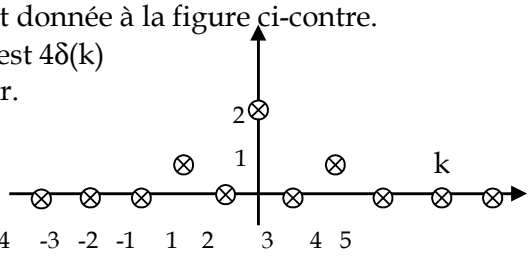
- Donner l'expression de Rx(k) et Sx(f) puis donner l'expression de Ry(k) et tracer la pour b_1=1 et b_2=-1.
- Connaissant la DSP du signal y(n), sur quoi se base-t-on pour le choix du modèle ?

3. Soit un processus AR défini par : $y(n)=-a_1y(n-1) - a_2\, y(n-2) + x(n)$ où x(n) bb décorrélé de variance 1

- Calculer $\mu_y(n)$ puis sans calcul, expliquer pourquoi y(n) est SSL.
- Montrer que pour k>0, $Ry(k)=-a_1Ry(k-1)-a_2\, Ry(k-2)$
- Déterminer $a_1$et a_2

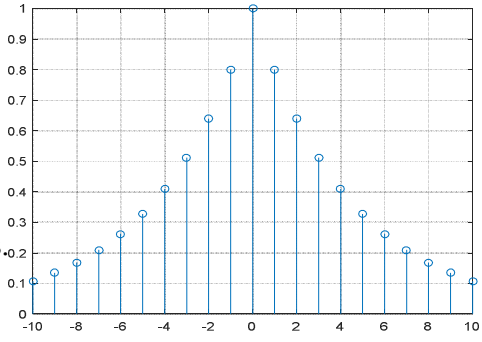
4. On veut modéliser le signal y(n) dont l'autocorrélation statistique est donnée à la figure ci-contre. Ry(k). On suppose que l'autocorrélation statistique du signal d'entrée est 4δ(k)

- Déterminer les moments statistiques d'ordre 1 du signal à modéliser.
- S'agit-il d'un modèle AR ou MA? Justifier
- Déterminer l'ordre du modèle
- En supposant que 2 coefficients sont égaux, à partir des valeurs de Ry(k), déduire que l'un des coefficients est nul puis déterminer les coefficients de ce modèle puis donner l'équation aux différences liant y(n) et x(n)



5. On veut modéliser le signal y(n) SSL dont la corrélation statistique n'a été tracée que de -10 à 10. Elle est donnée à la figure ci-contre.

- S'agit-il d'un modèle AR ou MA? Justifier
- Donner les propriétés statistiques du signal d'entrée.
- On suppose que le modèle est d'ordre 1, déterminer ses paramètres.
- Expliquer la notion de filtre formeur et son utilité en téléphonie



Solutions

1. Modèle AR ($R_y(k)\neq 0$) x(n) entrée bb, y(n) signal aléatoire à modéliser h(n) filtre formeur (modèle math)entrée bb + ergodisme $a_1 = -\alpha$ $\sigma_x^2 = 1-\alpha^2$

2. $R_x(k)=\sigma_x^2\delta(k)$ $S_x(f)=\sigma_x^2$ MA d'ordre 2 $R_y(0)=1+b_1^2+b_2^2 R_y(1)=b_1(1+b_2)$ $R_y(2)= b_2$ $R_y(k)= 0$ pour $k\geq 0$

3. $\mu_y(n)=0$ entrée bb SSL ⇒sortie SSL Interro1 14/15 **4.** Voir examen 17/18 **5.** Examen 16/17

Exercices supplémentaires

1.On considère un signal aléatoire stationnaire x(n) et l'on suppose connu ses coefficients d'autocorrélation : $R(0) = 3\sigma^2$, $R(1) = 2\sigma^2$, $R(2) = \sigma^2$, $R(3) = 0$

- On cherche le filtre MA d'ordre 3 de ce signal. Identifiez les paramètres du filtre.
- Si le signal x(n) a été obtenu par filtrage d'un bruit blanc gaussien de variance σ^2 par un filtre à réponse impulsionnelle finie de fonction de transfert $H(z) = 1+ a.z^{-1} + b.z^{-2}$, en déduire les valeurs de a et b.

2. Soit un filtre formeur dont l'équation aux différences est y(n)=0.5 (x(n)+x(n-1)+x(n-2)+x(n-3))

- Expliquer la notion de filtre formeur puis identifier ce modèle linéaire AR ou MA
- Donner la moyenne, l'autocorrélation et la DSP de son entrée *x(n).*
- Calculer et tracer $R_{yy}(k)$ puis déterminer $S_{yy}(f)$

3. Les signaux x(n) et y(n) ont été obtenus en filtrant, au moyen d'un filtre à réponse impulsionnelle finie, un bruit blanc b(n) gaussien centré de variance σ^2.

x(n) = 2.b(n) + 0.5.b(n -1) -0.2.b(n - 2) + 0.1.b(n - 3)y(n) = b(n) - b(n - 2)

- Identifier les 2 modèles puis pour chacun, calculer et tracer les coefficients d'autocorrélation
- x(n) et y(n) sont-ils Gaussiens (justifier)
- Calculer les intercorrélations Rxy(k), et Ryx(k) et commenter
- Donner quelques applications des modèles AR,MA, ARMA

4. Soit un filtre formeur dont l'équation aux différences est $y(n)= - \alpha y(n-1) - \beta y(n-2) + x(n)$

- Identifier l'ordre du modèle linéaire AR.
- Déterminer les paramètres du modèles, on suppose que $R_{yy}(k)=2*0.5^{|k|}$

5. Soit un filtre formeur dont l'équation aux différences est $y(n)= \alpha y(n-1) + x(n)$

- Identifier l'ordre du modèle linéaire AR.
- Déterminer la moyenne de y(n) et montrer que $R_{yy}(k)=\alpha^k R_{yy}(0)$, déduire une condition sur α.
- Montrer que $R_{yy}(0)=R_{xx}(0)/(1-\alpha^2)$
- Tracer $R_{yy}(k)$ (Prendre $R_{xx}(0)=\sigma^2=1$).
- Citer 2 applications concrètes des modèles AR.

6. Soit un filtre formeur dont l'équation aux différences est y(n)= 0.25 y(n-1) - 0.25 y(n-2) + x(n)

- Identifier ce modèle linéaire AR ou MA (Justifier)
- Déterminer la moyenne de y(n) et donner l'expression de $R_{yy}(k)$.
- Calculer et tracer $R_{yy}(k)$ (Prendre $R_{xx}(0)=\sigma^2=1$).

Solutions

1. $b_0=1$ $b_1=1$ et $b_2=1$ etpar identification, on trouve a=1 et b=1

2. MA, $\mu x=0$ et $Sx(f)=\sigma^2 Ry(0)=\sigma^2$, $Ry(1)=Ry(-1)=0.75\ \sigma^2$, $Ry(-2)=Ry(2)=0.5\ \sigma^2$, $Ry(-3)=Ry(3)=0.25\ \sigma^2$, $Ry(k>3)=0.Sy(f)=\sigma^2\ (1+1.5\cos(4\pi f)+ \cos(6\pi f) + 0.5\cos(8\pi f))$

3. Interro2 15/16 **4.** Ratt 15/16 **5.** Examen 15/16 **6.** Ratt 14/15

TP n° 4 : Modélisation paramétrique et Filtrage Numérique adaptatif

Ce TP a pour objectif :
- Modéliser un signal aléatoire par les modèles AR et MA (par approximation AR) et d'en extraire les informations utiles.
- Tester une méthode de filtrage adaptatif (LMS)

Exercice 1: Télécharger le fichier 'vous avez du courrier en attente.wav' et le placer dans le même répertoire que votre programme puis sélectionner une partie entre 21000 et 21500

I. Démo

```
# -*- coding: utf-8 -*-

""" Modélisation AR """
# import numpy as np; import matplotlib.pyplot as plt;
# from scipy.io import wavfile as wf;
# import scipy.linalg as la; import scipy.signal as sp;

# fname = 'vousavezducourrierenattente.wav';
# #winsound.PlaySound(fname, winsound.SND_FILENAME)
# fe, S = wf.read(fname);
# Te=1/fe; N=len(S); t = np.linspace(0, N-1, N)*Te;
# plt.figure(1);plt.subplot(211); plt.plot(t,S); plt.grid(True);
# plt.xlabel('temps'); plt.ylabel('Amp'); plt.title('Phrase');
# L=500;N1=21000;N2=N1+L; y=S[N1:N2]  ; ty=np.linspace(N1, N2-1, N2-N1)*Te;
# plt.subplot(212); plt.plot(ty,y); plt.grid(True);
# plt.xlabel('temps'); plt.ylabel('Amp'); plt.title('morceau de la phrase');


# """ Modélisation AR"""
# P=20;          "P Nbre de coefficicents du modèle AR>2"
# y=y-np.mean(y)
# R = np.correlate(y,y,mode='full')[len(y)-1:];
# #R=R/R.max();  "Autocor"
# C = la.toeplitz(R[0:P]);
# B = -R[1:P+1];
# a = la.inv(C).dot(B)
# a = np.append(1,a)
# sigma_carre = sum(a*R[0:P+1]);
# """ Visualisation de la réponse fréquentielle du filtre trouvé et du signal y de départ"""
# b = np.array([1,0]);
# f,H= sp.freqz(b,a,L//2,fs=fe);
# TFy = np.fft.fft(y); TFy = TFy[1:L//2+1];
# plt.figure(2); plt.subplot(121);plt.plot(f, 20*np.log10(abs(TFy/TFy.max())));
# plt.plot(f, 20*np.log10(abs(H/H.max())));
# plt.title('Module du Filtre trouvé (orange)+ Signal original modélisé(bleu)');
# plt.xlabel('Fréq (Hz)');plt.ylabel('Amp');
```

```
# # """ Synthèse à partir du filtre trouvé"""
# bb = (sigma_carre**0.5)*np.random.randn(L);
# #bb=bb-np.mean(bb);
# y_synt = sp.lfilter(b,a,bb) ;
# TFys = np.fft.fft(y_synt); TFys = TFys[0:L//2];
# plt.figure(2); plt.subplot(122);plt.plot(f, 20*np.log10(abs(TFys/TFys.max())));
# plt.plot(f, 20*np.log10(abs(TFy/TFy.max())));
# plt.title('signal synthétisé (orange)+ Signal synthétisé(bleu)');
# plt.xlabel('Fréq (Hz)');plt.ylabel('Amp');
"""-----------------------------------------------------------------"""

""" LMS : Algorithme du Gradient stochastique : Restauration """
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sp

fecg=[]; mecg=[];
""" Lecture des fihiers """
with open('fecg.txt') as f:
   for i in f: fecg.append(float(i))
fecg = np.array(fecg)
with open('mecg.txt') as f:
   for i in f : mecg.append(float(i))
mecg = np.array(mecg)
""" Rajout du bruit """
N=len(mecg); var = 0.01;
bb=(var**0.5)*np.random.randn(N);
x=fecg+bb+mecg; #x=x-np.mean(x)
y=fecg; #y=y-np.mean(y)

""" Détermination du filtre par LMS """
Ncoef=7; mu=0.1;
h = np.zeros(Ncoef) # Initialisation des coeffcients du filtre
L = len(x) #  Longueur du signal
yest = np.zeros(N) # y(n) estimé
En =  np.zeros(N) # Erreur
for i in range(Ncoef,L//10):
   X= np.flipud(x[i-Ncoef+1:i+1]);  #Prendre X=[x(n),x(n-1),...,x(n-Ncoef+1)))]
   yest[i] = np.dot(h, X)
   En[i] = x[i] - yest[i]
   h = h + mu * En[i] * X
h = h/sum(abs(h))

""" Visualisation des résultats"""
plt.figure(1)
y_est =sp.lfilter(h,1,x)
plt.subplot(311); plt.plot(x,'r', label= 'observation'); plt.plot(y,'b', label= 'fecg');plt.grid(); plt.legend()
plt.subplot(313);plt.plot(En,'g',label='erreur'); plt.grid(); plt.legend()
plt.subplot(312); plt.plot(y,'b', label= 'fecg'); plt.plot(y_est,'r', label= 'ecg restauré'); plt.legend()
"""-----------------------------------------------------------------"""
```

```
# """ Prédiction par LMS """
# mux=0; varx=0.01; N=1000
# bb=mux+np.sqrt(varx)*np.random.randn(N)

# fe=10000 ;Te=1/fe;
# f0= 100;
# t = np.arange(0,N)*Te
# y =  0.5*np.cos(2*np.pi*f0*t)
# x = y + bb

# Ncoef=5; mu=0.2;
# hp = np.zeros(Ncoef) # Initialisation des coeffcients du filtre
# L = len(x) #  Longueur du signal
# yest = np.zeros(N) # y(n) estimé
# En =  np.zeros(N) # Erreur
# for i in range(Ncoef,L-1):
#     X= np.flipud(x[i-Ncoef+1:i+1]);  #Prendre X=[x(n),x(n-1),...,x(n-Ncoef+1)))]
#     yest[i] = np.dot(hp, X)
#     En[i] = x[i] - yest[i]
#     hp = hp + mu * En[i] * X

# plt.figure(2); plt.plot(x,'b', label= 'observation'); plt.plot(yest,'r', label= 'estimé')
# plt.plot(En,'g', label= 'Erreur instantanée'); plt.grid(); plt.legend()
"""----------------------------------------------------------------------"""


""" Identification de système par LMS """
# mux=0; varx=0.1; N=500
# x = mux+np.sqrt(varx)*np.random.randn(N)
# b = np.array([0.65, -0.35, 0.1])
# y = sp.lfilter(b,[1,0],x)

# Ncoef=3; mu=0.6;
# hs = np.zeros(Ncoef) # Initialisation des coeffcients du filtre
# yest = np.zeros(N) # y(n) estimé
# En =  np.zeros(N) # Erreur
# for i in range(Ncoef,N-1):
#     X= np.flipud(x[i-Ncoef+1:i+1]);  #Prendre X=[x(n),x(n-1),...,x(n-Ncoef+1)))]
#     yest[i] = np.dot(hs, X)
#     En[i] = y[i] - yest[i]
#     hs = hs + mu * En[i] * X

# print(hs)
# plt.figure(3); plt.plot(y,'b', label= 'Sortie'); plt.plot(yest,'r', label= 'Sortie estimée')
# plt.plot(En,'g', label= 'Erreur instantanée'); plt.grid(); plt.legend()
```

```
"""A faire AR """

# import numpy as np; import scipy.signal as sp; import matplotlib.pyplot as plt
# import sounddevice as snd
# import scipy.io.wavfile as wav
# import scipy.linalg as la;

# def ModeleAR(y,P):
#     y=y-np.mean(y)
#     R = np.correlate(y,y,mode='full')[len(y)-1:]
#     C = la.toeplitz(R[0:P])
#     B = -R[1:P+1]
#     a = la.inv(C).dot(B)
#     a = np.append(1,a)
#     sigma_carre = sum(a*R[0:P+1]);
#     return a,sigma_carre

# fe,x = wav.read('voyelles.wav')
# snd.play(x, fe)

# 1. Lire, afficher et écouter le fichier voyelles.wav
# 2. Repérer les 3 parties correspondants aux 3 "A"
# NA=7000
# A1 = x[162400:162400+NA];
# A2 = x[253200:253200+NA]
# A3 = x[340000:340000+NA]
# 3. Afficher les 3 "A"
# 4. Calculer et afficher leur periodogramme en db
# 5. Déterminer le filtre formeur de chaque "A" en utilisant la foncton ModeleAR
# 6. Calculer l'erreur quadratique moyenne (MSE) entre les coefficents de chaque A
# from sklearn.metrics import mean_squared_error
# mseA12=mean_squared_error(a1,a2)
# 7. Refaire les étape de 2 à 6 pour la lettre "E"
# 8. Calculer le MSE entre les 3 "A" et les 3 "E"

"""A faire Restauration par LMS """
"""
# 1. Lire l'audio vousavezducourrierenattente.wav dabns une variable y
# 2. Créer un signal x composé du signal audio (y) + du bruit blanc
# 3. Ecouter pour vérifier la présence du bruit
# 4. Centrer les variables y et x
# 5. Déterminer par LMS le filtre h en prenant la partie correspondant au silence (mu=0.01 et for i in range(Ncoef,100) )
# 6. Inverser et normaliser le filtre h
# 7. Filtrer x par le filre
# 8. Visualiser les résultats
# 9. Ecouter l'audio restauré
# ss=input()
# snd.play(np.int8(y_est), fe)
"""
```