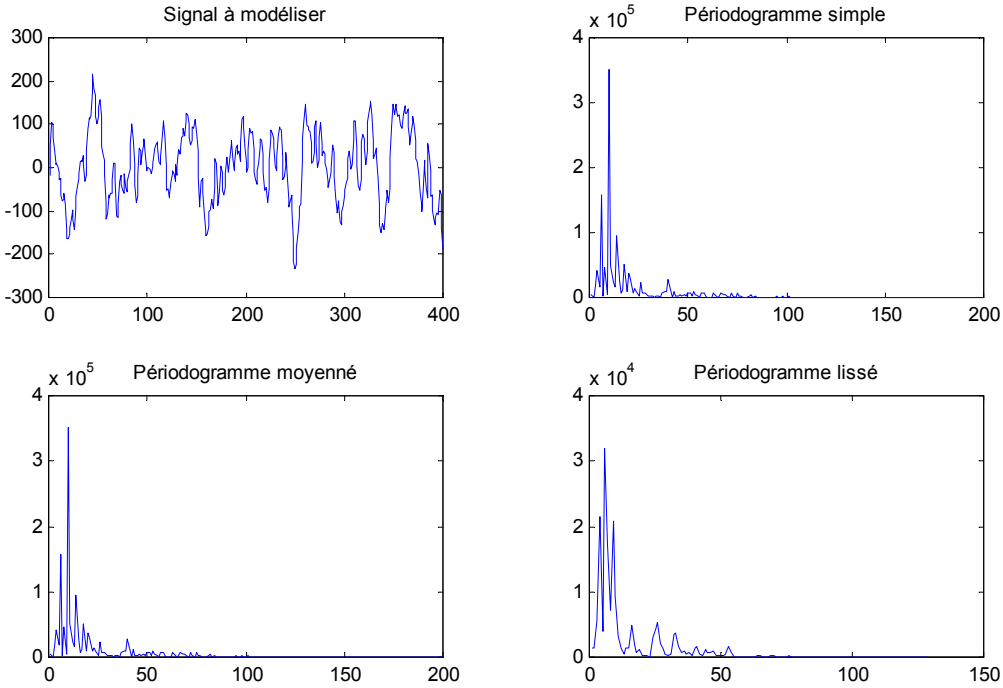
ANALYSE SPECTRALE - MÉTHODES PARAMÉTRIQUES

Les méthodes paramétriques d'analyse spectrale nécessitent une connaissance a priori sur les caractéristiques du signal (allure générale de son spectre). Elles permettent de dresser un modèle mathématique du signal à partir des coefficients d'un filtre. Pour illustrer ces méthodes, un signal d'électroencéphalogramme (EEG) va être analysé puis modélisé.

PREMIÈRE ANALYSE

Le signal est, dans un premier temps, analysé avec des méthodes non paramétriques (périodogrammes) permettant d'avoir l'a priori nécessaire pour le modéliser.



Analyse non paramétrique du signal EEG

Cette première analyse permet de déterminer les méthodes paramétriques à utiliser. Ce signal est constitué de composante fréquentielles dominantes (pics fréquentiels). On peut dores et déjà

prévoir qu'un modèle *autorégressif* (AR) permettra de modéliser au mieux ce signal. Le modèle *moving average* (MA) étant plus adapté à la modélisation de signaux dont le spectre est assez plat, son utilisation sur un tel signal ne donnera pas de bons résultats. Ces deux modèles paramétriques vont être testés sur le signal EEG afin de vérifier ces affirmations.

ANALYSE PARAMÉTRIQUE

La modélisation d'un signal par des techniques paramétriques nécessite une intervention humaine sur le modèle. Il faut, dans un premier temps, définir le modèle le mieux adapté au signal à modéliser. Une fois le modèle choisi, il faut déterminer son ordre, c'est à dire le nombre de coefficients permettant au mieux de modéliser ce signal. Le choix du modèle se fait en général en fonction de l'allure du spectre, mais l'ordre du modèle ne peut pas être déterminé précisément sans une analyse plus fine. Le choix de l'ordre se fait en général en minimisant un certain critère d'erreur entre le signal en sortie du modèle et le signal mesuré.

Pour chacun des deux modèles étudiés, quatre critères d'erreur sont minimisés, les critères d'*Akaike*, *MDL*, *Final Prediction Error* et *Parzen*. Ces quatre critères sont ne sont pas uniques, chaque cas peut nécessiter la minimisation selon un critère personnel en fonction des contraintes.

Modèle autorégressif (AR)

Principe

Un modèle autorégressif est un filtre *tout-pôle* (composé uniquement de pôles) au travers duquel passe un bruit blanc. On détermine les coefficients de ce filtre tels que le signal à sa sortie soit le plus proche possible du signal à modéliser.

Fonction de transfert d'un filtre autorégressif d'ordre P :

$$H(z) = \frac{1}{1 + \sum_{k=1}^{P} a_p(k) \cdot z^{-k}}$$

La sortie d'un tel filtre dépend des échantillons précédents de la sortie et de l'échantillon actuel du bruit. Les coefficients $a_p(k)$ agissent uniquement sur les échantillons précédents de la sortie.

Fonction Matlab

La fonction suivante calcule les coefficients d'un modèle autorégressif en minimisant l'un des quatre critères donnés précédemment. La méthode de calcul est la résolution de l'équation de Yule Walker par l'algorithme de Levinson. Trois modes sont possibles :

- Dans le premier mode, on appelle la fonction en lui donnant uniquement les échantillons du signal à modéliser. La fonction étudie l'évolution des différents critères et indique l'ordre idéal pour chacun. Elle renvoie tous les coefficients calculés sans tenir compte de la minimisation. Ce mode permet de voir les différences entre chaque critère.
- Dans le second mode, on appelle la fonction en lui donnant les échantillons du signal ainsi que le critère à minimiser. La fonction minimise le critère et renvoie les coefficients du modèle associés à ce critère.
- Dans le troisième mode, on appelle la fonction en lui donnant les échantillons du signal ainsi que l'ordre souhaité pour le modèle. Aucune minimisation d'erreur est faite, les coefficients sont directement renvoyés.

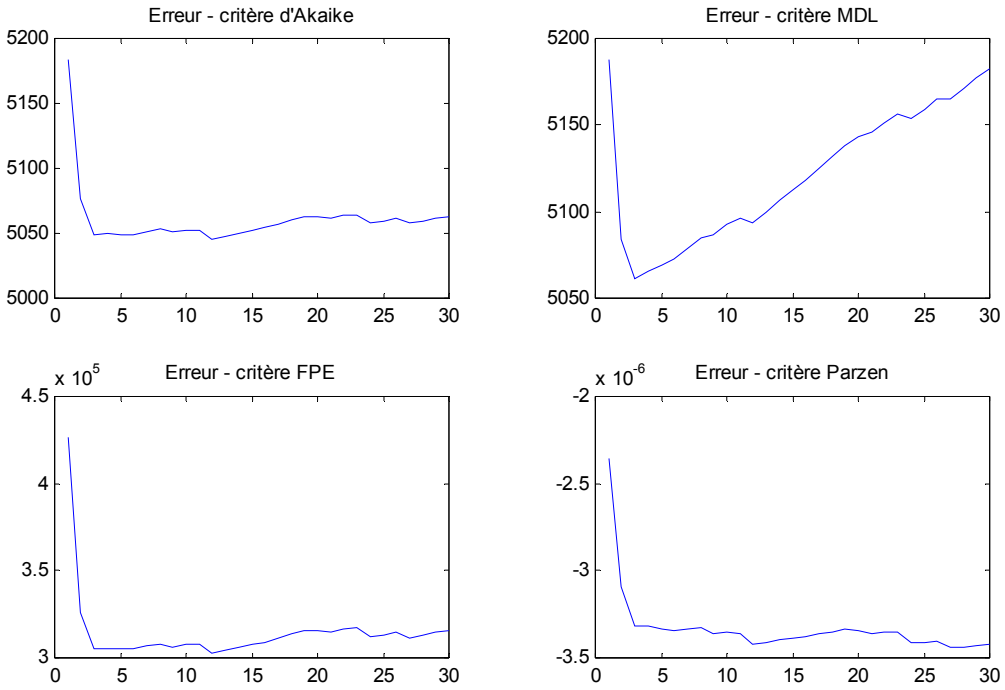
```
% coeffs = AR(s, critere, ordre)
% Calcule les paramètres d'un modèle autorégressif
%
% Arguments :
%    s = Signal à modéliser
%    critère = Critère d'erreur à optimiser (optionnel)
%          valeurs : 'Akaike', 'MDL', 'FPE', 'Parzen'
%    ordre = Ordre du modèle (optionnel)
%    coeffs = Coefficients du modèle
%
% Exemple :
%    Voir TP n°5 de traitement de signal
%
%                              (c) SeB + Cosmin 2002

function coeffs = AR(s, critere, ordre)
```

```
nb_iter = 30;
N = length(s);

if nargin == 1
   critere = '';
   ordre = -1;
end
if nargin == 2
   if(ischar(critere))
      ordre = -1;
   else
      ordre = critere;
   end
end

if( ordre == -1)
     for k=1 : nb_iter
           % Calcul des coefficients
           R = xcorr(s,k+1,'biased');
           R = toeplitz(R(k+2:end));
           A = R(1:k,1:k);
           B = -R(2:k+1,1);
           coeffs = inv(A)*B;
           coeffs = [1 coeffs'];

           % Calcul de l'erreur
           ErrTmp = s - filter([0 -coeffs(2:end)],1,s);
           ErrTmp = sum(ErrTmp.^2);
           erreur(k) = ErrTmp;
           akaike(k) = N*log(ErrTmp) + 2*k;
           MDL(k) = N*log(ErrTmp) + log(N)*k;
           FPE(k) = ErrTmp * (N+k+1)/(N-k-1);

           ParzTmp = 0;
           for j=1 : k
                 ParzTmp = ParzTmp + (N-j)/(N*erreur(j));
                 ParzTmp = ParzTmp / N;
                 ParzTmp = ParzTmp - (N-k)/(N*ErrTmp);
           end
           parzen(k) = ParzTmp;
     end

     switch(critere)
           case 'Akaike'
                 figure
                 plot(akaike)
                 title('Erreur - critère d''Akaike')
                 disp('Akaike - ordre optimal :')
                 disp(find(akaike == min(akaike)))
                 coeffs = coeffs(1:find(akaike == min(akaike))+1);
           case 'MDL'
                 figure
                 plot(MDL)
                 title('Erreur - critère MDL')
                 disp('MDL - ordre optimal :')
                 disp(find(MDL == min(MDL)))
                 coeffs = coeffs(1:find(MDL == min(MDL))+1);
           case 'FPE'
                 figure
                 plot(FPE)
                 title('Erreur - critère FPE')
                 disp('FPE - ordre optimal :')
                 disp(find(FPE == min(FPE)))
                 coeffs = coeffs(1:find(FPE == min(FPE)));
           case 'Parzen'
                 figure
                 plot(parzen)
                 title('Erreur - critère Parzen')
                 disp('Parzen - ordre optimal :')
```

```
                    disp(find(parzen == min(parzen)))
                    coeffs = coeffs(1:find(parzen == min(parzen)));
               case ''
                    figure
                    Subplot(2,2,1)
                    plot(akaike)
                    title('Erreur - critère d''Akaike')
                    disp('Akaike - ordre optimal :')
                    disp(find(akaike == min(akaike)))

                    Subplot(2,2,2)
                    plot(MDL)
                    title('Erreur - critère MDL')
                    disp('MDL - ordre optimal :')
                    disp(find(MDL == min(MDL)))

                    Subplot(2,2,3)
                    plot(FPE)
                    title('Erreur - critère FPE')
                    disp('FPE - ordre optimal :')
                    disp(find(FPE == min(FPE)))

                    Subplot(2,2,4)
                    plot(parzen)
                    title('Erreur - critère Parzen')
                    disp('Parzen - ordre optimal :')
                    disp(find(parzen == min(parzen)))
          end
     else
          k = ordre;
          % Calcul des coefficients
          R = xcorr(s,k+1,'biased');
          R = toeplitz(R(k+2:end));
          A = R(1:k,1:k);
          B = -R(2:k+1,1);
          coeffs = inv(A)*B;
          coeffs = [1 coeffs'];
     end


     return
```

Minimisation de l'erreur, choix de l'ordre

L'évolution de l'erreur est étudiée selon les quatre critères (premier mode de fonctionnement de la fonction).



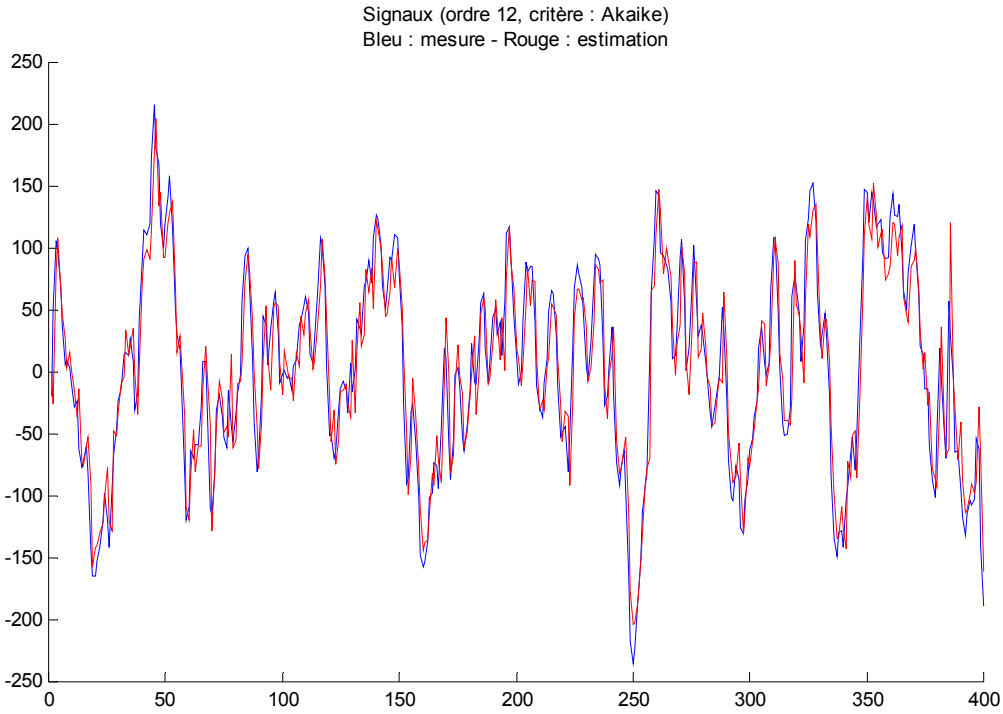
Évolution de l'erreur

On remarque que, selon le critère choisi, l'erreur n'évolue pas de la même façon (la courbe est plus ou moins plate). On remarque également que l'ordre optimal pour le modèle varie d'un critère à l'autre.

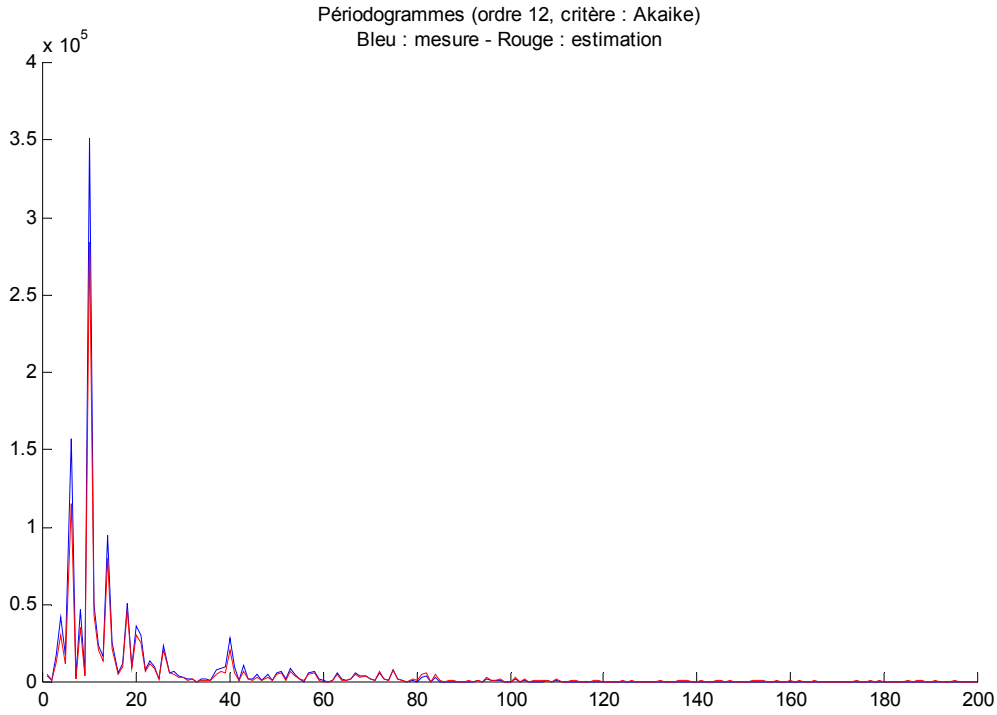
```
Akaike - ordre optimal :
    12

MDL - ordre optimal :
     3

FPE - ordre optimal :
    12

Parzen - ordre optimal :
    27
```

Modélisation du signal EEG

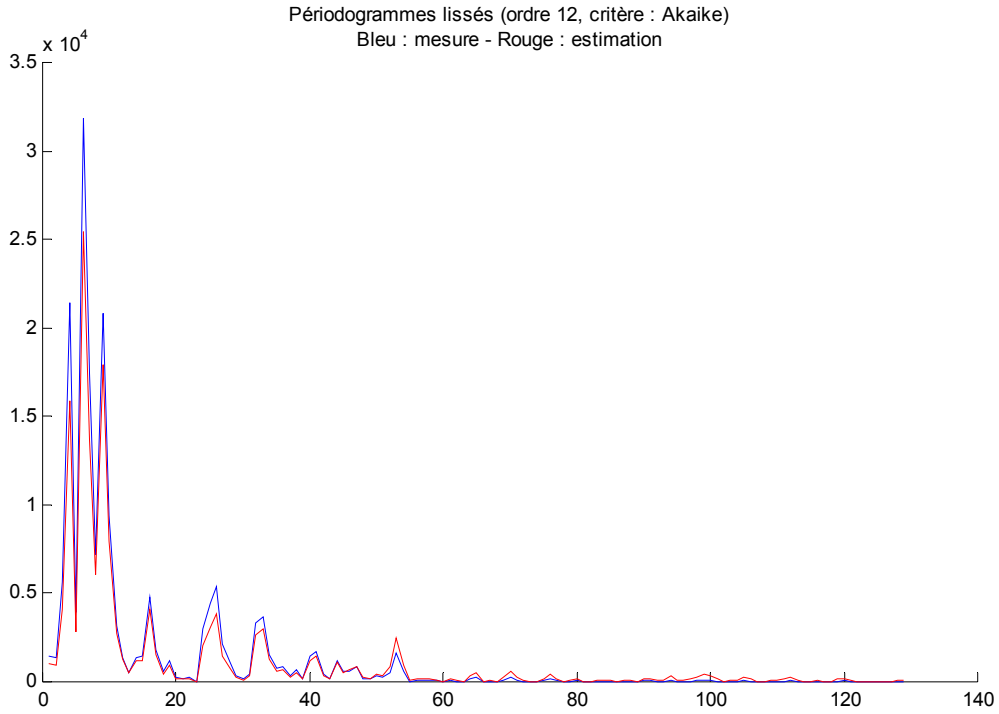
Pour la visualisation du comportement du modèle, le critère d'erreur choisi est le critère d'*Akaike*. L'ordre du modèle autorégressif est donc 12, le modèle comporte 13 coefficients.



Représentation temporelle des signaux



Représentation fréquentielle des signaux



Représentation fréquentielle lissée des signaux

On remarque assez facilement que les deux signaux (la mesure et l'estimation par le modèle autorégressif) sont quasiment confondus. L'a priori sur le résultat du modèle autorégressif est donc vérifié : pour ce type de signaux (à composantes fréquentielles dominantes), le modèle autorégressif est satisfaisant.

Modèle moving average (MA)

Principe

Un modèle moving average est un filtre *tout-zéro* (composé uniquement de zéros) au travers duquel passe un bruit blanc. On détermine les coefficients de ce filtre tels que le signal à sa sortie soit le plus proche possible du signal à modéliser.

Fonction de transfert d'un filtre moving average d'ordre P :

$$H(z) = \sum_{k=1}^{P} b_p(k) \cdot z^{-k}$$

La sortie d'un tel filtre dépend des échantillons précédents du bruit. Les coefficients $b_p(k)$ agissent uniquement ces derniers.

Fonction Matlab

La fonction suivante calcule les coefficients d'un modèle moving average en minimisant l'un des quatre critères donnés précédemment. Trois modes sont possibles :

- Dans le premier mode, on appelle la fonction en lui donnant uniquement les échantillons du signal à modéliser. La fonction étudie l'évolution des différents critères et indique l'ordre idéal pour chacun. Elle renvoie tous les coefficients calculés sans tenir compte de la minimisation. Ce mode permet de voir les différences entre chaque critère.
- Dans le second mode, on appelle la fonction en lui donnant les échantillons du signal ainsi que le critère à minimiser. La fonction minimise le critère et renvoie les coefficients du modèle associés à ce critère.
- Dans le troisième mode, on appelle la fonction en lui donnant les échantillons du signal ainsi que l'ordre souhaité pour le modèle. Aucune minimisation d'erreur est faite, les coefficients sont directement renvoyés.

Le calcul des coefficients du modèle se fait par l'intermédiaire de la fonction 'ar' donnée précédemment. La technique utilisée consiste à modéliser le signal comme un modèle autorégressif d'ordre trois fois supérieur à l'ordre du modèle moving average souhaité. Les coefficients ainsi obtenus sont considérés comme un second signal que l'on modélise comme un modèle autorégressif d'ordre égal à celui souhaité. Les coefficients obtenus sont ceux du modèle moving average recherché.

Si on veut modéliser un signal S par un modèle moving average d'ordre 4, on commence par calculer les coefficients d'un modèle autorégressif d'ordre 12 du signal S. On obtient donc les coefficients $\{ a_1 , \dots , a_{12} \}$ que l'on considère comme les échantillons d'un signal S'. On calcule alors les coefficients du modèle autorégressif d'ordre 4 du signal S'. On obtient les coefficients $\{ b_1 , \dots , b_4 \}$ du modèle moving average du signal S.

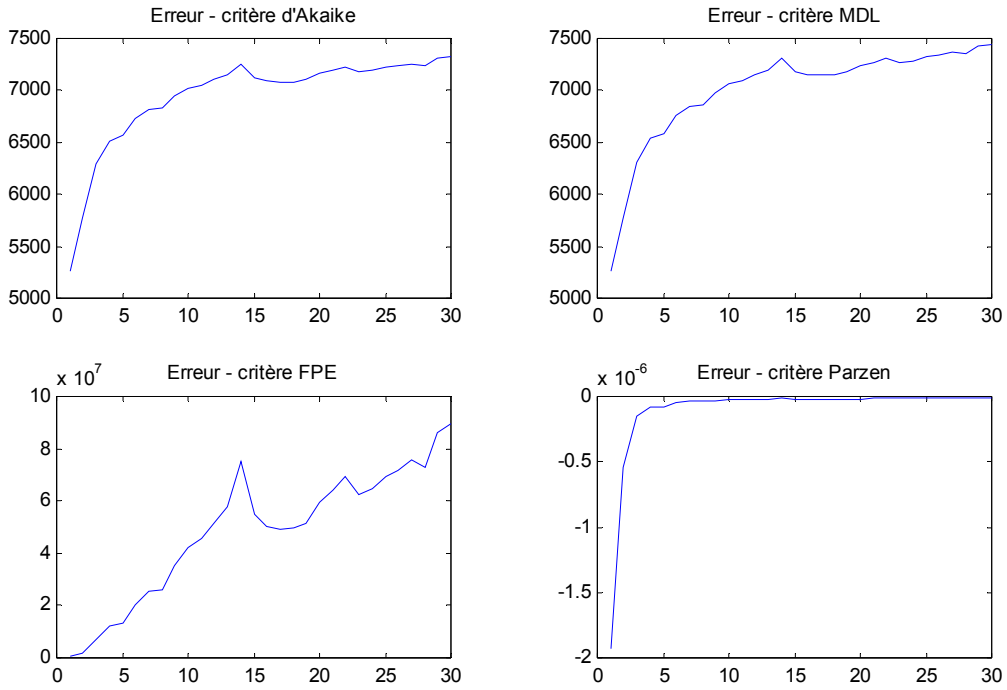
```
% coeffs = MA(s, critere, ordre)
%
% Calcule les paramètres d'un modèle Moving Average
%
% Arguments :
%    s = Signal à modéliser
%    critère = Critère d'erreur à optimiser (optionnel)
```

```
%          valeurs : 'Akaike', 'MDL', 'FPE', 'Parzen'
%    ordre = Ordre du modèle (optionnel)
%    coeffs = Coefficients du modèle
%
% Exemple :
%    Voir TP n°5 de traitement de signal
%
%                              (c) SeB + Cosmin 2002

function coeffs = MA(s, critere, ordre)

nb_iter = 30;
N = length(s);

if nargin == 1
   critere = '';
   ordre = -1;
end
if nargin == 2
   if(ischar(critere))
      ordre = -1;
   else
      ordre = critere;
   end
end

if( ordre == -1)
     for k=1 : nb_iter
           c1 = AR(s,3*k);
           coeffs = AR(c1,k);

           % Calcul de l'erreur
           ErrTmp = s - filter([0 coeffs(2:end)],1,s);
           ErrTmp = sum(ErrTmp.^2);
           erreur(k) = ErrTmp;
           akaike(k) = N*log(ErrTmp) + 2*k;
           MDL(k) = N*log(ErrTmp) + log(N)*k;
           FPE(k) = ErrTmp * (N+k+1)/(N-k-1);

           ParzTmp = 0;
           for j=1 : k
                 ParzTmp = ParzTmp + (N-j)/(N*erreur(j));
                 ParzTmp = ParzTmp / N;
                 ParzTmp = ParzTmp - (N-k)/(N*ErrTmp);
           end
           parzen(k) = ParzTmp;
     end

     switch(critere)
           case 'Akaike'
                 figure
                 plot(akaike)
                 title('Erreur - critère d''Akaike')
                 disp('Akaike - ordre optimal :')
                 disp(find(akaike == min(akaike)))
                 coeffs = coeffs(1:find(akaike == min(akaike))+1);
           case 'MDL'
                 figure
                 plot(MDL)
                 title('Erreur - critère MDL')
                 disp('MDL - ordre optimal :')
                 disp(find(MDL == min(MDL)))
                 coeffs = coeffs(1:find(MDL == min(MDL))+1);
           case 'FPE'
                 figure
                 plot(FPE)
                 title('Erreur - critère FPE')
                 disp('FPE - ordre optimal :')
                 disp(find(FPE == min(FPE)))
                 coeffs = coeffs(1:find(FPE == min(FPE)));
```

```
                case 'Parzen'
                      figure
                      plot(parzen)
                      title('Erreur - critère Parzen')
                      disp('Parzen - ordre optimal :')
                      disp(find(parzen == min(parzen)))
                      coeffs = coeffs(1:find(parzen == min(parzen)));
                case ''
                      figure
                      subplot(2,2,1)
                      plot(akaike)
                      title('Erreur - critère d''Akaike')
                      disp('Akaike - ordre optimal :')
                      disp(find(akaike == min(akaike)))

                      subplot(2,2,2)
                      plot(MDL)
                      title('Erreur - critère MDL')
                      disp('MDL - ordre optimal :')
                      disp(find(MDL == min(MDL)))

                      subplot(2,2,3)
                      plot(FPE)
                      title('Erreur - critère FPE')
                      disp('FPE - ordre optimal :')
                      disp(find(FPE == min(FPE)))

                      subplot(2,2,4)
                      plot(parzen)
                      title('Erreur - critère Parzen')
                      disp('Parzen - ordre optimal :')
                      disp(find(parzen == min(parzen)))
          end
     else
          k = ordre;
          c1 = AR(s,3*k);
          coeffs = AR(c1,k);
     end

     return
```

Minimisation de l'erreur, choix de l'ordre

L'évolution de l'erreur est étudiée selon les quatre critères (premier mode de fonctionnement de la fonction).

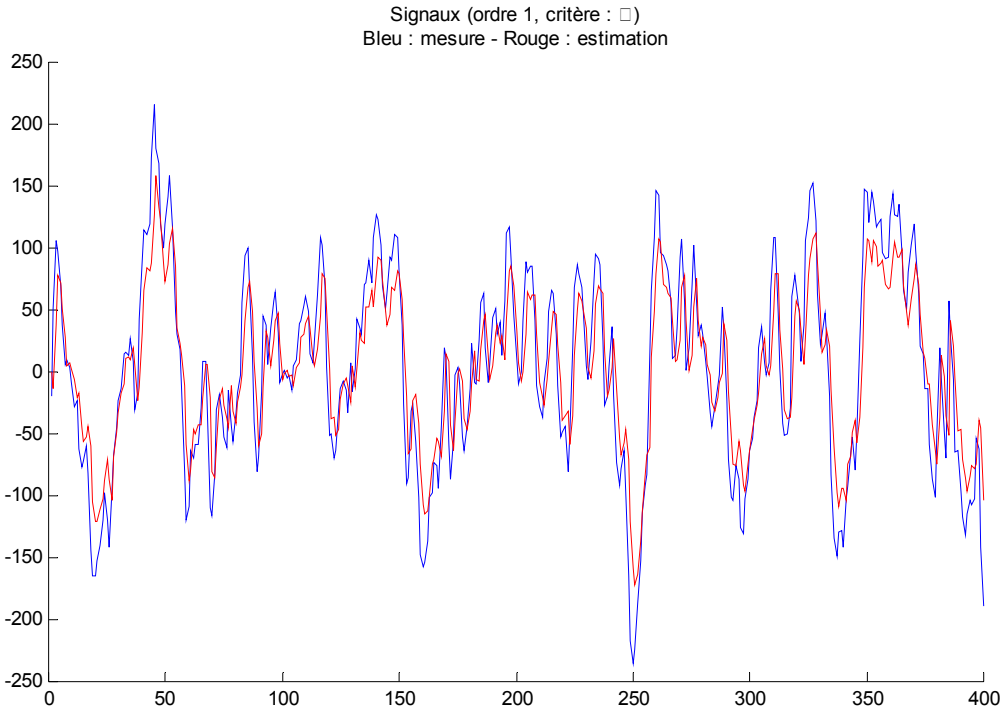


Évolution de l'erreur

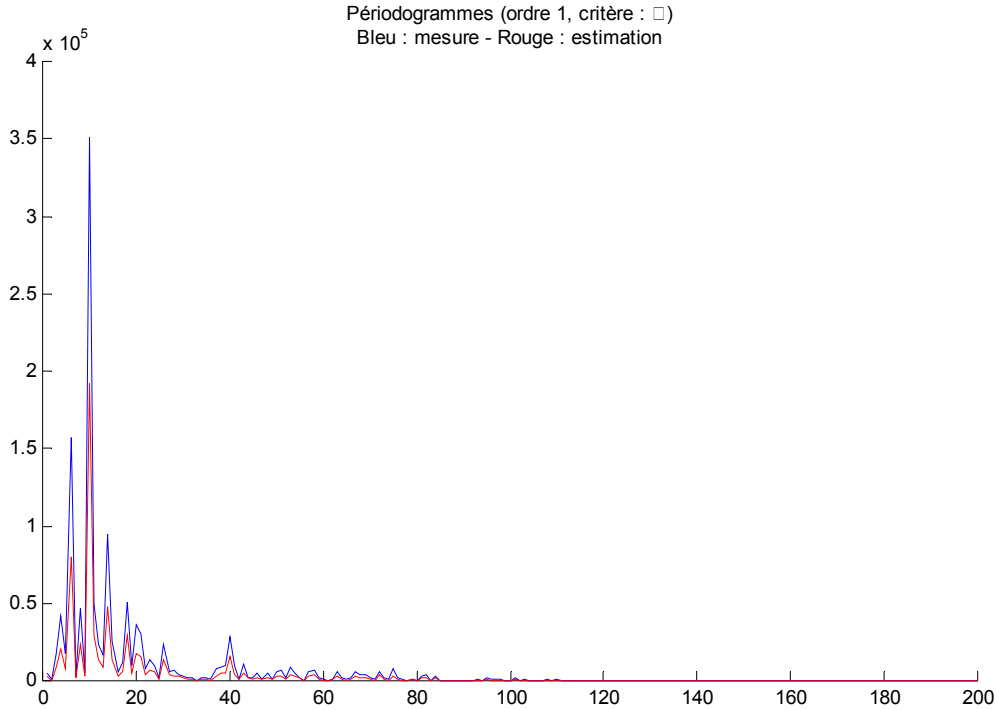
On remarque que pour ce type de signaux, l'erreur diverge fortement. Pour chacun de ces critère, l'ordre optimal est le même, mais l'erreur est plus importante que pour le modèle autorégressif.

```
Akaike - ordre optimal :
     1

MDL - ordre optimal :
     1

FPE - ordre optimal :
     1

Parzen - ordre optimal :
     1
```

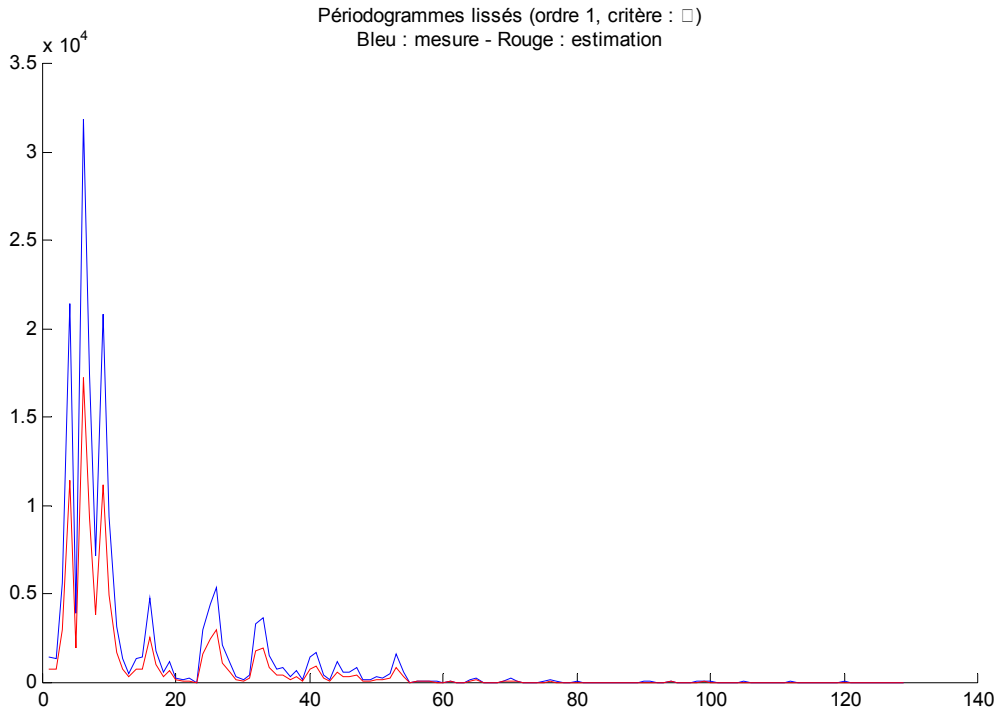
Modélisation du signal EEG



Représentation temporelle des signaux



Représentation fréquentielle des signaux



Représentation fréquentielle lissée des signaux

CONCLUSION

Ces techniques paramétriques sont très utiles dans les cas où le signal a des caractéristiques connues, souvent fréquentielles. Elles permettent une estimation assez simple et plutôt efficace du signal sans forcément en connaître la source et les modifications à y apporter. Le fait d'ignorer la provenance du signal permet de supposer qu'il a été généré par un bruit blanc passé à travers un filtre. Cette supposition permet de modéliser n'importe quel signal sans restriction, à condition de posséder quelques indications sur l'allure de son spectre. Elle offre également l'avantage d'être applicable à n'importe quel processus physique assimilable à un filtre.