TP 2 : Estimation de direction d'arrivée par corrélation, synthèse de filtres analogiques et numériques RII

Il est indispensable d'avoir préparé le TP avant la séance, c'est-à-dire :
- **avoir lu le sujet de TP et avoir compris le cours et les travaux dirigés correspondants ;**
- **avoir étudié le document « Matlab pour le traitement du signal » ;**
- **maîtriser les commandes Matlab utilisées durant la première séance de TP.**

Les listings de tous les programmes doivent être joints au rapport.

Tous les résultats et courbes doivent être commentés.

Présentation du TP

L'objectif de cette manipulation est d'étudier le principe de d'estimation de direction d'arrivée d'un signal audio à partir de deux microphones, typiquement placés sur un robot. On a vu en Travaux Dirigés qu'il était possible de détecter cette direction d'arrivée à partir de l'intercorrélation entre les deux signaux. C'est ce que l'on va mettre en œuvre ici dans un cas simple puis dans le cas plus défavorable de signaux bruités. Dans le cas de signaux bruités, il est parfois possible de réduire le bruit par filtrage, en synthétisant un filtre adapté au problème.

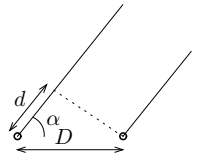
Tout naturellement, votre travail se décompose en trois étapes :
- l'étude de la corrélation et l'estimation de la direction d'arrivée à partir des signaux non bruités ;
- l'analyse du signal bruité à filtrer et la détermination des caractéristiques du filtre ;
- la synthèse du filtre ;
- le filtrage du signal et l'analyse du signal filtré ;
- l'estimation de la direction d'arrivée à partir des signaux débruités.

Concernant le filtrage, nous utiliserons ici un filtre numérique à réponse impulsionnelle infinie (RII) que nous construirons par discrétisation d'un filtre analogique par transformation bilinéaire .

Avant d'arriver en séance, vous devez impérativement avoir assimilé les notions théoriques concernant :
- l'estimation de direction d'arrivée par corrélation de signaux ;
- le filtrage des signaux et la représentation spectrale de cette opération ;
- la synthèse des filtres numériques RII par transformation bilinéaire.

I Estimation de direction du locuteur par corrélation

Un robot possède deux microphones séparés d'une distance $D = 25$ cm lui permettant d'acquérir des signaux stéréo. En analysant les signaux d'une personne qui lui parle, il peut estimer l'angle α entre cette personne et l'axe de ses microphones et donc se tourner vers elle. En effet, un des signaux ayant une distance plus grande à parcourir, il est décalé temporellement par rapport à l'autre : $x_2(t) = x_1(t-\tau)$. Il suffit donc au robot d'estimer le retard τ entre les deux signaux, ce qui lui permet alors de calculer la distance $d = \tau \cdot c$, où $c = 340$ m.s^{-1} est la vitesse de propagation du son, et d'en déduire l'angle $\alpha = \text{acos}(d/D)$ (voir le schéma ci-contre).



Tous les signaux manipulés durant ce TP ont été enregistrés avec la même direction d'arrivée. Vous disposez dans le fichier `stereo.wav` d'un tel signal stéréo et vous allez effectuer un programme permettant d'estimer cet angle α. Lorsque vous chargez le signal correspondant à l'aide de la fonction Matlab `[x, Fe] = wavread('stereo.wav');`, vous obtenez une matrice à deux colonnes, chaque colonne correspondant à l'un des signaux (canaux droit et gauche).

1. Rappeler la relation entre les transformées de Fourier $\widehat{x}_1(f)$ et $\widehat{x}_2(f)$. Plus précisément, donner les relations entre les modules de ces TF et les relations entre leurs phases.
2. Rappeler la relation existant entre l'intercorrélation $C_{x_1,x_2}(t)$ des deux signaux et l'autocorrélation du signal $C_{x_1}(t)$. Comment peut-on calculer le décalage entre les deux signaux à partir de cette intercorrélation $C_{x_1,x_2}(t)$?
   En pratique, on travaillera avec des signaux à temps discret et les corrélations $C_{x_1,x_2}[k]$ et $C_{x_1}[k]$ seront calculées pour des décalages k entiers (appelés *lags* en Anglais).
3. Écouter le signal stéréo à l'aide de la fonction Matlab `PlaySignal(x,Fe)`. Pouvez-vous à l'oreille estimer la direction de la personne ?
4. Tracer sur une même courbe les représentations temporelles $x_1(t)$ et $x_2(t)$ de ces deux signaux. Peut-on aisément calculer le décalage entre ces deux signaux à partir de ces représentations ?
5. Tracer les représentations fréquentielles $\widehat{x}_1(f)$ et $\widehat{x}_2(f)$ de ces deux signaux[1], avec sur une courbe les modules en dB (`20*log10(abs(X))`) et sur une autre la phase (`angle(X)`). Peut-on aisément calculer le décalage entre ces deux signaux à partir de ces représentations ?
6. Calculer l'autocorrélation $C_{x_1}[k]$ et l'intercorrélation $C_{x_1,x_2}[k]$ à l'aide de la fonction Matlab `xcorr` (`[Cx1x2,lags] = xcorr(x1,x2);` où `lags` sont les indices de décalages k, voir annexe). Tracer sur une même courbe, pour des décalages temporels en secondes, l'intercorrélation $C_{x_1,x_2}[k]$ et l'autocorrélation $C_{x_1}[k]$.
7. La fonction Matlab `max` permet de calculer la valeur maximale `val` d'un vecteur et l'indice `ind` correspondant (`[val,ind] = max(vect);`). Calculer le décalage entre les deux signaux grâce à l'intercorrélation $C_{x_1,x_2}[k]$ et en déduire l'angle entre le robot et la personne.
8. Le signal précédent était relativement idéal puisque les deux signaux n'étaient pas perturbés. Refaire l'étude précédente avec le signal perturbé contenu dans le fichier `stereo_bruit1.wav`. Que constatez-vous ?

1. Vous pourrez pour cela, comme au TP précédent, calculer la transformée de Fourier discrète des échantillons du signal que vous tracerez en fonction de la fréquence.

II Filtrage pour améliorer la détection de direction du locuteur par corrélation

Le cas bruité précédent était très favorable d'un point de vue traitement du signal car le bruit était blanc (contenait toutes les fréquences avec la même puissance moyenne) et décorrélé du signal de parole. Dans des situations moins favorables, l'estimation est perturbée par d'autres signaux environnants plus gênants. Cependant, on peut parfois procéder par un simple filtrage pour éliminer les signaux perturbateurs. C'est ce que l'on va faire ici...

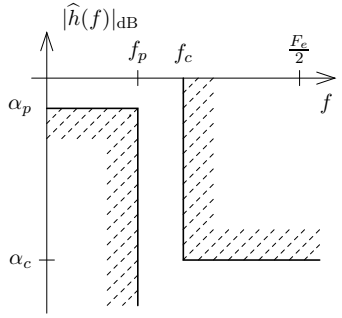
Le signal contenu dans le fichier `stereo_bruit2.wav` est un cas typique d'un tel problème

II.1 Analyse du signal et rédaction du cahier des charges du filtre

1. Écouter le signal bruité. Entend-on clairement la perturbation ? Comment la caractériseriez-vous ?
2. Arrivez-vous à retrouver l'angle entre le robot et la personne à partir de ce signal ?
3. Tracez la représentation temporelle du signal bruité et la comparer à celle du signal non bruité. Voit-on clairement la perturbation ?
4. Tracez la représentation fréquentielle du signal bruité en échelle linéaire et logarithmique (c'est-à-dire en dB) et la comparer à celle du signal non bruité. Voit-on clairement la perturbation ? Quel type de filtre proposez-vous de synthétiser afin de la supprimer ?

Afin d'éliminer la perturbation, on va filtrer le signal avec un filtre passe-bas dont il faut préciser le Gabarit. On requiert que les caractéristiques fréquentielles du filtre soient les suivantes :

- la bande coupée $[f_c, F_e/2]$ doit correspondre à la bande dans lequel les perturbations sont plus importantes que le signal.
- la bande passante $[0, f_p]$ doit correspondre à la bande dans lequel le signal est majoritairement présent.
- la bande de transition entre les bandes passantes et coupées $]f_p, f_c[$ doit être d'une largeur suffisante pour que l'ordre du filtre ne soit pas trop important ;
- l'atténuation en bande passante doit être inférieure à 1 dB (soit $\alpha_p = -1$) ;
- l'atténuation en bande coupée doit être raisonnablement grande au vu de l'analyse du signal, tout en gardant un ordre du filtre raisonnable.



5. A partir de ces caractéristiques et de la représentation fréquentielle du signal bruité, déterminez les valeurs des paramètres du gabarit du filtre numérique à synthétiser. Commenter votre choix tant en fréquence qu'en atténuation...

Ce gabarit définira le cahier des charges que vous devrez respecter dans la partie suivante. **Il est fortement recommandé, dans votre programme Matlab,** de définir les variables `fp`, `fc`, `alphap` et `alphac` et de faire la synthèse des filtres à partir de ces variables. Ainsi, une simple modification de ces variables suffira à modifier votre filtre si nécessaire !

Vous allez synthétiser un filtre numérique répondant à ce cahier des charges en discrétisant un filtre analogique passe-bas de Butterworth par transformation bilinéaire.

II.2 Synthèse d'un filtre passe-bas de Butterworth

Nous allons synthétiser un filtre passe-bas numérique satisfaisant le gabarit en procédant en deux étapes : *1)* synthèse d'un filtre passe-bas analogique *2)* discrétisation de ce filtre par transformation bilinéaire. A partir de la fonction de transfert du filtre analogique $H_a(p)$, la transformation bilinéaire permet d'obtenir simplement la fonction de transfert du filtre numérique : $H_n(z) = H_a(\frac{2}{T_e}\frac{z-1}{z+1})$. Nous avons vu en cours qu'une telle méthode permettait de synthétiser un filtre numérique stable satisfaisant le gabarit **si l'on prenait soin de pré-déformer les fréquences** caractéristiques du gabarit.

1. A l'aide de la fonction `butterworth_lowpass` (voir annexe), soit avec la commande :
   ```
   [num_a, den_a] = butterworth_lowpass(fp,fc,alphap,alphac);
   ```
   calculez le numérateur `num_a` et le dénominateur `den_a` de la fonction de transfert du filtre analogique de Butterworth satisfaisant votre gabarit. Attention : si l'ordre de ce filtre est supérieur à 20 (`Ordre = length(den_a)-1`), prendre un gabarit moins strict (bande de transition plus large par exemple...)
2. Tracez la réponse en fréquence en dB du filtre analogique (fonction `freqs`, voir annexe), avec les commandes :
   ```
   P = 4096; freq = (0:P-1)/P*Fe/2; % Vecteur des fréquences
   H_a = freqs(num_a, den_a, 2*pi*freq); % Réponse en fréquences
   plot(freq,20*log10(abs(H_a)));
   ```
   (vous remarquerez que la fonction `freqs` travaille en pulsation plutôt qu'en fréquence). Superposer à cette courbe le gabarit (fonction `DisplayFilterSpecif`, voir annexe) avec les commandes :
   ```
   hold on; DisplayFilterSpecif(fp,fc,alphap,alphac,Fe/2); hold off;
   ```
   Cette réponse en fréquence satisfait-elle le gabarit ?
3. Calculez le numérateur et dénominateur du filtre numérique passe-bas à partir du filtre analogique passe-bas par transformation bilinéaire (fonction `bilinear`, voir annexe)
   ```
   [num_n,den_n] = bilinear(num_a,den_a,Fe); % transformation bilinéaire
   ```
4. Tracez la réponse en fréquence du filtre numérique (voir fonction `freqz` en annexe) à laquelle vous superposerez le gabarit, soit les commandes :
   ```
   H_n = freqz(num_n,den_n,P,Fe); % Réponse en fréquences
   plot(freq ,20*log10(abs(H_n)));
   DisplayFilterSpecif(fp,fc,alphap,alphac,Fe/2); hold off;
   ```
   La réponse en fréquence de ce filtre numérique satisfait-elle le gabarit ?
5. Modifier les fréquences caractéristiques du filtre analogique avec la pré-déformation vue en cours (pour cela, modifier uniquement les fréquences `fp` et `fc` dans l'appel à la fonction `butterworth_lowpass`). Synthétiser le filtre analogique correspondant à ses fréquences pré-déformées puis le filtre numérique correspondant par transformation bilinéaire. La réponse en fréquence de ce filtre numérique satisfait-elle le gabarit ?
6. Le filtre synthétisé est-il bien stable ?

II.3 Analyse du signal filtré

A l'aide du filtre numérique passe-bas satisfaisant le gabarit :

1. Filtrez le signal avec la fonction `filter` : `y = filter(num_n,den_n,x);` (voir annexe).
2. Tracez les représentations temporelles et fréquentielles du signal original `x` et du signal filtré `y`. Voit-on encore les perturbations dans le signal filtré ?
3. Écouter le signal filtré. Entend-on encore les perturbations ?
4. Estimer la direction d'arrivée à partir du signal filtré. Le résultat est-il effectivement amélioré ?
5. Dans un cadre général, que pensez-vous qu'il se passera si plusieurs signaux arrivent en même temps ?

III Annexe

Vous utiliserez lors de ce TP les fonctions Matlab `xcorr`, `butterworth_lowpass`, `freqs`, `DisplayFilterSpecif`, `bilinear`, `freqz`, `filter`. Le fonctionnement de certaines fonctions a été mentionné dans le document *Matlab et le traitement du signal*. L'aide simplifiée de ces fonctions est donnée ci-dessous, l'aide complète étant disponible sur Matlab (fonction `help`).

III.1 Fonction `xcorr`

```
 XCORR Cross-correlation function estimates.
    C = XCORR(A,B), where A and B are length M vectors (M>1), returns
    the length 2*M-1 cross-correlation sequence C. If A and B are of
    different length, the shortest one is zero-padded.  C will be a
    row vector if A is a row vector, and a column vector if A is a
    column vector.

    XCORR(...,MAXLAG) computes the (auto/cross) correlation over the
    range of lags:  -MAXLAG to MAXLAG, i.e., 2*MAXLAG+1 lags.
    If missing, default is MAXLAG = M-1.

    [C,LAGS] = XCORR(...)  returns a vector of lag indices (LAGS).
```

III.2 Fonction `butterworth_lowpass`

```
BUTTERWORTH_LOWPASS : Butterworth analog lowpass filter prototype.
   [NUM,DEN] = BUTTERWORTH_LOWPASS(f_p,f_c,alpha_p,alpha_c)
   forms the transfer function of a Butterworth analog lowpass filter
   from the characteristics f_p, f_c, alpha_p, alpha_c of its prototype
                    NUM(p)
            H(p) = --------
                    DEN(p)
   Vectors NUM and DEN are returned with numerator and denominator
   coefficients in descending powers of p
```


III.3 Fonction Matlab `freqs`

```
 freqs Laplace-transform (s-domain) frequency response.
    H = freqs(B,A,W) returns the complex frequency response vector H
    of the filter B/A:
                          nb-1         nb-2
            B(s)   b(1)s     +  b(2)s     + ... +  b(nb)
      H(s) = ---- = -----------------------------------
                          na-1         na-2
            A(s)   a(1)s     +  a(2)s     + ... +  a(na)
    given the numerator and denominator coefficients in vectors B and A.
    The frequency response is evaluated at the points specified in
    vector W (in rad/s).  The magnitude and phase can be graphed by
    calling freqs(B,A,W) with no output arguments.

    [H,W] = freqs(B,A) automatically picks a set of 200 frequencies W on
    which the frequency response is computed.  freqs(B,A,N) picks N
    frequencies.
```

III.4 Fonction `DisplayFilterSpecif`

```
 DisplayFilterSpecif(f_p,f_c,alpha_p,alpha_s[,fmax])
    Display the filter design specifications of a filter
    that loses no more than Rp dB in the passband and
    has at least Rs dB of attenuation in the stopband, for example
    alpha_p = -1, alpha_c = -40;
    f_p and f_c are the passband and stopband edge frequencies, for example,
        Lowpass:    f_p = .1,      f_c = .2
        Highpass:   f_p = .2,      f_c = .1
        Bandpass:   f_p = [.2 .7], f_c = [.1 .8]
        Bandstop:   f_p = [.1 .8], f_c = [.2 .7]
    fmax is the highest frequency of the graph
```

III.5 Fonction Matlab `bilinear`

```
 BILINEAR Bilinear transformation with optional frequency prewarping.
    [NUMd,DENd] = BILINEAR(NUM,DEN,Fs) converts the s-domain transfer
    function specified by NUM and DEN to a z-transform discrete
    equivalent obtained from the bilinear transformation:
        H(z) = H(p) |
                    | p = 2*Fs*(z-1)/(z+1)
     where Fs is the sample frequency in Hz.
```

III.6 Fonction Matlab `freqz`

```
FREQZ Digital filter frequency response.
   [H,W] = FREQZ(B,A,N) returns the N-point complex frequency response
   vector H and the N-point frequency vector W in radians/sample of
   the filter:  jw               -jw              -jmw
        jw  B(e)    b(1) + b(2)e + .... + b(m+1)e
     H(e) = ---- = ----------------------------------
               jw               -jw              -jnw
            A(e)    a(1) + a(2)e + .... + a(n+1)e
   given numerator and denominator coefficients in vectors B and A. The
   frequency response is evaluated at N points equally spaced around the
   upper half of the unit circle. If N isn't specified, it defaults to 512.

   [H,F] = FREQZ(B,A,N,Fs) and [H,F] = FREQZ(B,A,N,'whole',Fs) given a
   sampling freq. Fs in Hz return a frequency vector F in Hz.
```

Remarque importante : Pour afficher la réponse en fréquence sur `P` échantillons (`P` quelconque) d'un filtre numérique de réponse en fréquence définie par les polynômes `B` (numérateur) et `A` (dénominateur) pour des fréquences de `0` à la moitié de la fréquence d'échantillonnage `Fe` il suffit donc écrire :

```
[H, freq] = freqz(B,A,P,Fe);
plot(freq,20*log10(abs(H)));
```

III.7 Fonction Matlab `filter`

```
FILTER One-dimensional digital filter.
   Y = FILTER(B,A,X) filters the data in vector X with the
   filter described by vectors A and B to create the filtered
   data Y.  The filter is a "Direct Form II Transposed"
   implementation of the standard difference equation:

   a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                         - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```