TP3 : Analyse, codage et synthèse de la parole

** Ce TP est fortement basé sur les cours et les TD associés. Il est donc indispensable de réviser les cours et les TD avant la séance.*
** Les questions marquées du signe † doivent être résolues avant la séance.*

I. Analyse

Le fichier *sig.wav* contient un signal de parole correspondant à l'enregistrement du mot « assécher », échantillonné à Fe=11 kHz. Charger ce signal dans Matlab en utilisant la fonction *audioread* et l'écouter.

1. Tracer la représentation temporelle (audiogramme) de ce signal. Pouvez-vous distinguer les 5 phonèmes ? Créer un nouveau signal *x* en supprimant les zones de silence avant et après le mot dans le signal d'origine. En zoomant sur chaque phonème, vérifier s'il s'agit d'un son voisé on non-voisé. Délimiter chaque phonème.

2. Tracer le module de la transformée de Fourier du signal *x* et commenter le résultat.

3. Tracer le spectrogramme à bande étroite du signal (calculé sur les tranches de 0.05 sec) en utilisant la fonction Matlab ci-dessous :

```
spectrogram(x,0.05*Fe,0.025*Fe,0.05*Fe,Fe,'yaxis');
```

Peut-on estimer approximativement la période pitch du signal à partir de ce spectrogramme ?

4. Tracer le spectrogramme à bande large du signal (calculé sur les tranches de 0.01 sec) en utilisant la fonction Matlab ci-dessous :

```
spectrogram(x,0.01*Fe,0.005*Fe,0.01*Fe,Fe,'yaxis');
```

Peut-on distinguer les 5 phonèmes sur ce spectrogramme ? Peut-on en conclure la similitude entre le troisième et le cinquième phonèmes ?

II. Codage LPC

Dans la suite, on divise le signal *x* en tranches de 30 millisecondes (non-recouvrantes) et on estime les paramètres du modèle LPC de la production du signal de parole pour ces tranches.

Pour décider si une tranche du signal est voisée ou non-voisée, on peut en pratique se baser sur des critères tels que la puissance moyenne ou le taux de passages par zéro. Pour gagner du temps, on n'utilisera pas ce genre de critère dans la suite. On se basera plutôt sur nos informations a priori : si la tranche du signal appartient aux 2ème ou 4ème phonèmes, elle est non-voisée, sinon elle est voisée.

5. Estimer la puissance moyenne de chaque tranche et stocker le résultat dans un vecteur.

6. Si la tranche est voisée, estimer sa période pitch en estimant la fonction d'autocorrélation de la tranche avec la version biaisée de la fonction Matlab *xcorr* et en identifiant la position du pic de cette fonction dans l'intervalle des valeurs possibles de la période pitch (de 1/600 à 1/70 sec). Si la tranche est non-voisée, on associera une valeur nulle au pitch. Stocker le résultat dans un vecteur.

Le modèle autorégressif de signal de parole est défini par l'équation ci-dessous :

$$x(n)+a_1x(n-1)+a_2x(n-2)+\ldots+a_Px(n-P)=e(n)$$

7.† Quelle est la fonction de transfert de ce modèle ?

8.† En suivant l'approche présentée en TD, montrer que les coefficients a_i dans le modèle ci-dessus (appelés *coefficients LPC*) peuvent être estimés en résolvant l'équation matricielle de la page suivante où les échantillons de la fonction d'autocorrélation pourraient être calculés par la fonction Matlab *xcorr*. La matrice étant une matrice de Toeplitz (à diagonales constantes), elle peut être construite à partir de sa première ligne à l'aide de la fonction Matlab *toeplitz*.

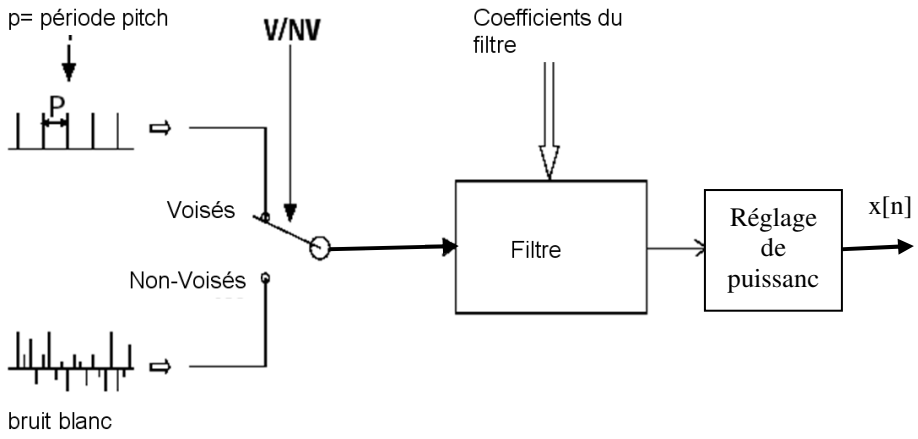
$$\begin{pmatrix} R_x(0) & R_x(1) & \cdots & R_x(P-1) \\ R_x(1) & R_x(0) & \cdots & R_x(P-2) \\ \vdots & \vdots & \cdots & \vdots \\ R_x(P-1) & R_x(P-2) & \cdots & R_x(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_P \end{pmatrix} = \begin{pmatrix} -R_x(1) \\ -R_x(2) \\ \vdots \\ -R_x(P) \end{pmatrix}$$

9. Calculer les coefficients LPC de chaque tranche pour un modèle d'ordre P=12 et stocker le résultat dans une matrice.

10. Si la puissance moyenne, la période pitch et chacun des coefficients LPC sont codés sur 8 bits, il faut combien de bits pour transmettre toutes les informations du signal entier ? Comparer avec le nombre de bits nécessaires pour envoyer le signal sans codage si chaque échantillon est codé sur 8 bits. Conclusion ?

III. Synthèse

On utilise le modèle ci-dessous pour synthétiser chaque tranche du signal de parole à partir des informations extraites dans la section précédente. Dans ce modèle, la source est un train d'impulsions de période pitch si la tranche est voisée et un bruit blanc (généré par la fonction Matlab *randn*) si elle est non-voisée. Le filtrage du signal source par le modèle autorégressif peut être effectué en utilisant la fonction Matlab *filter*.



11. Synthétiser le signal et écouter le résultat. Conclusion ?

12. Recommencer la synthèse en multipliant la période pitch de toutes les tranches par 2. Que constatez-vous ?

13. Recommencer en fixant une période pitch constante pour toutes les tranches voisées. Conclure sur l'importance de l'information pitch.

14. A partir de votre réponse à la question 7, calculer la réponse en fréquence du modèle AR. Si on suppose que $e(n)$ est un bruit blanc (le cas des phonèmes non-voisés), quelle est l'expression de la DSP d'une tranche du signal $x(n)$? Utiliser les coefficients du modèle AR identifiés pour tracer la racine carrée de cette estimation de DSP (à un facteur d'échelle près) pour 2 tranches du signal appartenant aux 2 phonèmes non-voisés. Pour ce faire, vous pouvez utiliser la fonction freqz de Matlab.

IV. Améliorations

Dans les parties analyse et synthèse, on a utilisé des fenêtres rectangulaires non-recouvrantes pour diviser le signal en plusieurs tranches de courte durée, puis les reconcaténer à la fin. Pour améliorer la qualité du signal synthétisé, on utilise souvent des fenêtres de Hanning partiellement recouvrantes pour l'analyse et la synthèse.

15. Modifier votre programme en utilisant des fenêtres de Hanning de 30 millisecondes avec un recouvrement de 50% entre deux fenêtres successives. Ecouter le signal synthétisé. Est-il de meilleure qualité ? Pourquoi ? Quel est le prix à payer ?

Annexe : Fonctions Matlab utilisées

spectrogram Spectrogram using a Short-Time Fourier Transform (STFT).

S = spectrogram(X) returns the spectrogram of the signal specified b vector X in the matrix S. By default, X is divided into eight segments with 50% overlap, each segment is windowed with a Hamming window. The number of frequency points used to calculate the discrete Fourier transforms is equal to the maximum of 256 or the next power of two greater than the length of each segment of X.

S = spectrogram(X,WINDOW) when WINDOW is a vector, divides X into segments of length equal to the length of WINDOW, and then windows each segment with the vector specified in WINDOW. If WINDOW is an integer, X is divided into segments of length equal to that integer value, and a Hamming window of equal length is used. If WINDOW is not specified, the default is used.

S = spectrogram(X,WINDOW,NOVERLAP) NOVERLAP is the number of samples each segment of X overlaps. NOVERLAP must be an integer smaller than WINDOW if WINDOW is an integer. NOVERLAP must be an integer smaller than the length of WINDOW if WINDOW is a vector. If NOVERLAP is not specified, the default value is used to obtain a 50% overlap.

S = spectrogram(X,WINDOW,NOVERLAP,NFFT) specifies the number of frequency points used to calculate the discrete Fourier transforms. If NFFT is not specified, the default NFFT is used.

S = spectrogram(X,WINDOW,NOVERLAP,NFFT,Fs) Fs is the sampling frequency specified in Hz. If Fs is specified as empty, it defaults to 1 Hz. If it is not specified, normalized frequency is used.

spectrogram(...) with no output arguments plots the PSD estimate for each segment on a surface in the current figure.

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:

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

freqz Frequency response of digital filter.

[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:

$$H(e^{jw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1) + b(2)e^{-jw} + ... + b(m+1)e^{-jmw}}{a(1) + a(2)e^{-jw} + ... + a(n+1)e^{-jnw}}$$

given numerator and denominator coefficients in vectors B and A.

[H,F] = freqz(...,N,Fs) and [H,F] = freqz(...,N,'whole',Fs) return frequency vector F (in Hz), where Fs is the sampling frequency (in Hz).

freqz(...) with no output arguments plots the magnitude and unwrapped phase of the filter in the current figure window.