ESIEE

UNITE : OBL-4101 **Projet : Vocodeur de phase**

OBJECTIFS DU PROJET ET ATTENDU DU RAPPORT

Ce projet a pour objectif de vous faire faire une réalisation en autonomie.

Les principales différences par rapport aux séances de TP sont :

- En plus de comprendre les « scripts » fournis, vous devez créer votre propre programme réalisant notamment la fonction principale du vocodeur de phase
- Vous n'avez plus juste à relier les observations à vos connaissances théoriques sur un thème précis mais vous devrez également penser, expliquer, justifier votre réalisation en fonction de vos connaissances et d'une bibliographie.
- Vous devrez rédiger un rapport complet. A lui seul, il devrait permettre au lecteur de comprendre l'objectif de votre travail, avoir les explications et justifications théoriques qui permettent de comprendre votre démarche pour la réalisation du vocodeur de phase, la réalisation (principe des algorithmes utilisés pour toutes les fonctions, y compris celles qui sont fournies) et, bien entendu, les résultats obtenus commentés.
  Vous ne devez donc pas vous contenter de répondre aux questions du sujet. Ce dernier n'est là que pour vous guider et pour vous permettre de ne pas « passer à côté » de points importants.
- Vous êtes libre de faire des propositions, des tests, des suggestions, proposer des traitements supplémentaires, des améliorations…

En conclusion, vous devez vous approprier le projet : vous ne répondez pas à des questions mais vous présentez votre projet.
Il est fortement conseillé de lire les consignes disponibles sur Blackboard

TRAVAIL A REALISER

Des programmes Matlab (fichiers avec l'extension .m) vous sont fournis.
Les commandes principales sont données dans ces programmes commentés. Vous devez bien entendu les comprendre et pouvez éventuellement les compléter/modifier (par exemple si vous souhaitez ajouter une figure, modifier des paramètres…).

Vous aurez à réaliser 2 fonctions principales :
1- le principe du vocodeur de phase → interpolation dans le domaine fréquentiel
2- la robotisation de la voix

RENDU DU TRAVAIL

Vous travaillerez en binôme. Chaque binôme déposera **son dossier (.zip)** sur Blackboard dans l'espace prévu pour cela (rubrique « Mon cours » → « Remise des rapports ») **en respectant son groupe**. Bien entendu, vous déposerez 1 seul rapport par binôme.
Aucun autre format ne sera corrigé : pas de zip7, pas de rar…

Ce fichier devra contenir **tous les programmes Matlab (fichiers .m)** nécessaires, **tous les fichiers audio (.wav)** utilisés ainsi que **le rapport sous format (.pdf)**. Attention, pas de script Matlab dans le rapport (nous aurons les programmes).

La date limite pour envoyer votre travail sera indiquée sur Blackboard.

ORGANISATION GENERALE DU PROGRAMME VOCODEUR DE PHASE

INTRODUCTION :

Le vocodeur consiste à « coder » de la voix. Le vocodeur de phase agira essentiellement sur la phase du signal. Attention, ici le terme de « codage » n'est pas pris au sens du codage de données mais au sens « modification ».

Nous allons voir 3 types de modifications de la voix :

- Le premier consiste à modifier la vitesse de la parole sans modifier le pitch. Le pitch étant lié à la fréquence fondamentale de la voix correspondant à la hauteur de la voix (grave, aigue…).
  On garde la « même voix » mais les mots seront prononcés plus lentement ou plus rapidement.

- Le deuxième consiste à modifier le pitch de la voix sans modifier la vitesse de prononciation.

- Enfin, le troisième va appliquer un effet de « robotisation » de la voix.

Un programme principal vous est fourni, « ***Vocodeur.m*** » permettant de lancer les 3 modifications. Vous pouvez modifier les paramètres, le compléter pour observer les signaux, tester sur d'autres fichiers de parole…

Structure (minimale) du programme principal :

1- Modification de la vitesse → appel de la fonction ***PVoc.m***
2- Modification du pitch → appel de la fonction ***PVoc.m*** + **ré-échantillonnage**
3- Robotisation de la voix → appel de la fonction ***Rob.m***

A : MODIFICATION DU PITCH, MODIFICATION DE LA VITESSE : « *PVOC.M* »

Pour les 2 premiers effets, la modification se fait dans le domaine fréquentiel en utilisant la transformée de Fourier à court terme (voir TP 1 pour le principe de la TFCT).

En effet, le signal de parole n'est pas strictement stationnaire. Par contre, on peut le considérer stationnaire sur une durée relativement faible (20 à 30 ms environ).
Nous utiliserons donc ses caractéristiques dans une fenêtre temporelle et feront la TF fenêtre par fenêtre. La fenêtre de pondération est une fenêtre de Hanning (vous pouvez voir la documentation de la fonction « hann » de Matlab).
Pour cela la fonction ***TFCT.m*** vous est fournie.

Nous appliquerons ensuite les modifications nécessaires à chaque portion (fenêtre) du signal fréquentiel. Vous allez alors créer la fonction que nous avons nommée « ***TFCT_Interp.m*** ».

Nous reviendrons ensuite dans le domaine temporel par une TF à court terme inverse : fonction fournie « ***TFCTInv.m*** ».

Ces 3 fonctions, *TFCT*, *TFCT_Interp* et *TFCTInv*, seront appelées dans l'ordre à partir d'une fonction plus générale « ***PVoc.m*** » fournie.

La seule différence entre la variation du pitch et la variation de la vitesse est que pour la variation du pitch, il ne faut pas oublier de ré-échantillonner le signal résultant pour avoir toujours 1 échantillon tous les *Te* et donc garder la même fréquence d'échantillonnage *Fe* (→ on garde la même « vitesse »). Pour cela, on utilise la fonction « ***resample*** » de Matlab dans le programme *Vocodeur.m*.

Le principal travail d'implémentation dans cette partie, est la réalisation du programme de la fonction *TFCT_Interp.m*. En vous aidant des explications données en cours, de votre recherche bibliographique et de l'annexe, vous devrez comprendre ce que doit réaliser cette fonction (et pourquoi) et, bien sûr, l'implémenter.

Remarque : *lorsque vous modifiez le pitch, vous pouvez également écouter (et observer) ce que cela donne en additionnant le résultat au signal d'origine… on peut faire des duos, des trios (test avec Extrait.wav par exemple)… on peut ajouter des « sopranos », « contre-ténors » ou « basses » dans une chorale qui n'en a pas (test avec Halleluia.wav par exemple)…*

B : ROBOTISATION DE LA VOIX : « *ROB.M* »

La robotisation de la voix permet d'obtenir un son synthétique largement utilisé dans le cinéma lorsque l'on veut faire « parler » des robots… d'où le terme « robotisation » de la voix. Cet effet est également largement utilisé en musique depuis quelques décennies déjà et qui revient ces dernières années notamment dans le RAP et le RNB (à ne pas confondre avec « l'autotune »).

Pour ce troisième effet, la modification se fait dans le domaine temporel et le principe est alors différent des 2 autres.
Cet effet sera réalisé par la fonction « ***Rob.m*** » que vous devez créer et à laquelle on fait appel dans le programme principal « *Vocodeur.m* ».

Vous devrez comprendre ce que doit réaliser cette fonction (et pourquoi) et, bien sûr, l'implémenter.

Une façon simple d'obtenir cet effet peut être réalisée directement dans le domaine temporel. On notera cependant qu'une méthode plus robuste consisterait à utiliser la transformée de Hilbert et de jouer sur la phase de chaque trame de la TFCT (nous ne détaillerons pas cela dans le cadre de ce cours).

Nous allons moduler 1 sinusoïde (ou plus exactement une exponentielle complexe) à une fréquence *fc* : $y_{rob}(t) = y(t).\exp(-j2\pi f_c.t)$. Comme le résultat est un signal complexe, nous devrons récupérer la partie réelle de ce dernier.

La valeur de *fc* déterminera le « degrés de robotisation ». Vous pourrez faire les tests avec *fc* = 2000, 1000, 500 et 200. Expliquez ce qu'il se passe pour chaque valeur de *fc*.

On notera que les choix de *fc* trop grand (2000 et 1000) ne sont pas judicieux en terme de son mais vous permettront de comprendre et d'expliquer ce qu'il se passe, notamment dans le domaine fréquentiel…
En fonction du résultat « sonore », vous pouvez proposer une autre valeur de *fc* qui vous semblerait plus judicieuse.

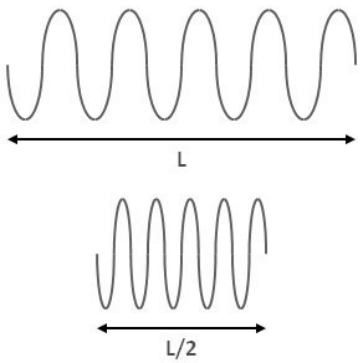
Remarque : *la robotisation est plus « judicieuse» sur un signal de parole plutôt que sur une voix chantée (tests sur Diner.wav par exemple). D'ailleurs ce serait bien de tester sur votre propre voix. Il suffit d'enregistrer un extrait de votre voix et d'appliquer la robotisation sur cet extrait…*

ANNEXE

A- PRINCIPE GENERAL DU VOCODEUR DE PHASE

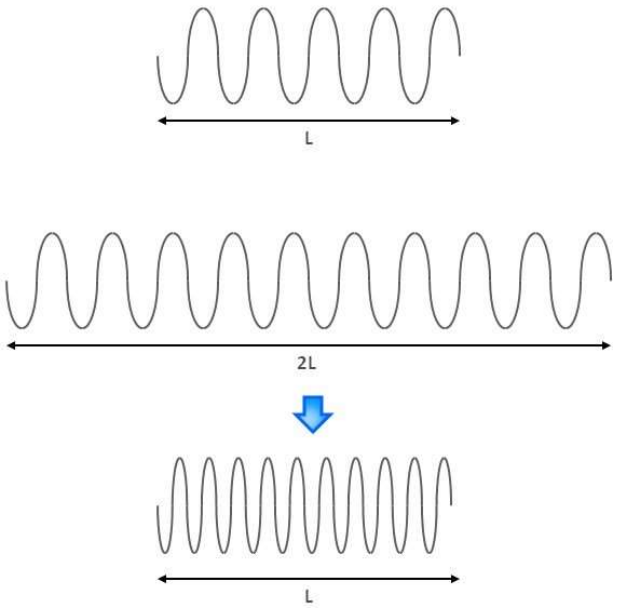
L'objectif est de « déplacer » le « pitch » d'un signal audio. Ce dernier étant définit par une fréquence fondamentale, le déplacement (ou transposition) se fait donc dans le domaine fréquentiel.

Une transposition directe de fréquence a pour effet de modifier la durée du signal.
Exemple sur une sinusoïde (figure ci-dessous) :
On considère une sinusoïde de fréquence f0 (courbe du haut). La transposition à une fréquence 2 fois plus élevée, 2.f0, donnerait la courbe du bas dont la durée est 2 fois plus courte (le son serait alors 2 fois plus « rapide »).



Ref. http://www.guitarpitchshifter.com/algorithm.html#33

Pour avoir l'impression d'avoir le même son mais prononcé (s'il s'agit de la parole) ou joué (s'il s'agit d'un instrument) par une voix ou un instrument d'une octave au-dessus, il faudrait d'abord interpoler pour avoir une durée 2 fois plus grand, puis transposer la fréquence (voir figure ci-dessous).
C'est ce que doit faire la fonction *TFCT_Interp.m* que vous devez créer.



Ref. http://www.guitarpitchshifter.com/algorithm.html#33

D'autre part, le passage dans le domaine fréquentiel ne peut se faire par une simple transformée de Fourier sur tous les échantillons. En effet, le signal n'est pas stationnaire sur l'ensemble de la durée. Par exemple, un signal de parole peut être considéré stationnaire sur une durée de 20 à 30 ms.
Dans cet intervalle, le pitch peut être considéré constant et nous pouvons lui appliquer la transposition. Nous devons donc le faire par tranches et chaque tranche sera appelée « trame ».
On utilise alors la transformée de Fourier à Court Terme (TFCT ou STFT en anglais) dont le principe a été abordé lors du premier TP.

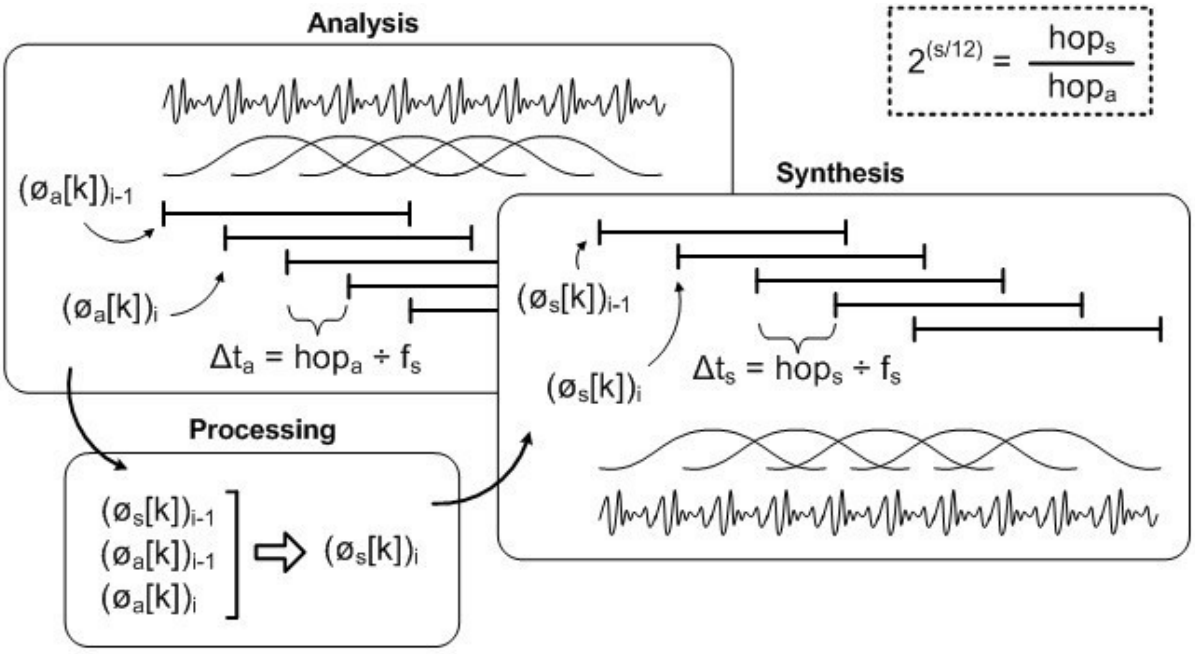
Trois étapes sont nécessaires dans le vocodeur de phase :

1- Le passage dans le domaine fréquentiel par TFCT correspondant à l'étape « analyse » → fonction Matlab fournie : *TFCT.m*

2- La transposition des trames correspondant à l'étape « traitement » → fonction Matlab à faire : *TFCT_Interp.m*

3- Le retour dans le domaine fréquentiel correspondant à l'étape « synthèse »→ fonction Matlab fournie : *TFCTInv.m*

On considère le schéma général ci-dessous :



Ref. http://www.guitarpitchshifter.com/algorithm.html#33

1- L'analyse → TFCT :

On rappelle que pour la TFCT, les trames ne sont pas juxtaposées mais se chevauchent sur un intervalle noté Nov dans les programmes et hop dans le schéma ci-dessus. D'autre part, avant d'appliquer la TF à une trame, on la pondère par une fenêtre. Nous avons choisi une fenêtre de Hanning.

Le résultat de TFCT sera donné non pas par un vecteur mais une matrice de N lignes et Ntr colonnes. N est le nombre de points de la TF pour chaque trame et Ntr est le nombre de trames. Chaque colonne représente donc la TF de chaque trame.
On notera que l'on ne garde que les échantillons aux fréquences positives de la TF (de 0 à Fe/2) → N = 1 + Nfft/2 si Nfft = N est le nombre de points de calcul de la TF.

2- Le traitement →l'interpolation dans le domaine fréquentiel :

Pour que le son reconstruit par TFCT inverse soit le plus fidèle possible au son original, il faut qu'en chaque fréquence la phase du signal modifié soit la même (ou la plus proche possible) du signal original.
En effet, l'information de phase entre 2 trames successives est très importante.
Pour chaque trame, le traitement devra donc faire une transposition (interpolation) du spectre d'amplitude (module de TF) et faire un traitement particulier de la phase (argument de la TF). D'où le terme de vocodeur de phase.

Voici le principe de la fonction que vous devez implémenter : *TFCT_Interp.m* :

La fonction débutera ainsi :

```
function y = TFCT_Interp(X,t,Nov)

% y = TFCT_Interp(X, t, hop)
% Interpolation du vecteur issu de la TFCT
%
% X : matrice issue de la TFCT
% t : vecteur des temps (valeurs réelles) sur lesquels on interpole
% Pour chaque valeur de t (et chaque colonne), on interpole le module du
spectre
% et on détermine la différence de phase entre 2 colonnes successives de X
%
% y : la sortie est une matrice où chaque colonne correspond à l'interpolation
de la colonne correspondante de X
% en préservant le saut de phase d'une colonne à l'autre
%
% Remarque : programme largement inspiré d'un programme fait à l'université de
Columbia


  [nl,nc] = size(X);    % récupération des dimensions de X

  N = 2*(nl-1);         % calcul de N (= Nfft en principe)


  % Initialisations
  %------------------------------------------------
  % Spectre interpolé
  y = zeros(nl, length(t));

  % Phase initiale
  phi = angle(X(:,1));

  % Déphasage entre chaque échantillon de la TF
  dphi0 = zeros(nl,1);  %initialisation
  dphi0(2:nl) = (2*pi*Nov)./(N./(1:(N/2)));

  % Premier indice de la colonne interpolée à calculer
  % (première colonne de Y). Cet indice sera incrémenté
  % dans la boucle
  Ncy = 1;

  % On ajoute à X une colonne de zéros pour éviter le problème de
  % X( : , Ncx2) en fin de boucle (Ncx2 peut être égal à nc+1)
  X = [X,zeros(nl,1)];
```

```
% Boucle pour l'interpolation
%-------------------------------------------------------------------
%Pour chaque valeur de t, on calcule la nouvelle colonne de Y à partir de 2
%colonnes successives de X
```
Votre algorithme à faire à partir d'ici !!!

Principe de l'algorithme (il s'agit d'indications) :

Chaque trame (colonne) de Y (numérotée par *Ncy*) est centrée à un instant *tn* du vecteur temps (t). Donc, pour chaque valeur de t notée (*tn*), on appliquera le traitement de 2 colonnes de X pour calculer 1 colonne de Y → il faut faire une boucle sur toutes les valeurs de t.

- Début de la boucle : pour chaque valeur *tn* de t
  - On récupère les 2 colonnes « adéquates » de X → il faut donc définir les indices *Ncx1* et *Ncx2* des 2 colonnes de X en fonction de la position de l'instant *tn* (rappelez-vous les explications en cours sur l'interpolation d'un point…)
  - On calcule le module *My* interpolé → voir explications ci-dessous
  - On calcule la colonne de Y : *Y(: , Ncy) = My .* exp(j*phi)*
  - On met à jour la nouvelle phase *phi* qui sera utilisée pour le calcul de la prochaine colonne de Y (prochaine trame) → voir explications ci-dessous (rappelez-vous l'importance de la continuité de la phase)
  - On incrémente l'indice colonne de Y : *Ncy = Ncy* + 1
- Fin de la boucle.

- Calcul des échantillons du module de la colonne (trame) interpolée numéro *Ncy* de Y (à l'instant *tn*) :

$$My = \alpha|X(:, Ncx1)| + \beta|X(:, Ncx2)|$$

avec :

$$\beta = tn - floor(tn)$$
$$\alpha = 1 - \beta$$

- Mise à jour de la phase :
  calcul de la variation de phase entre les 2 colonnes successives de X

$$d\varphi = angle(X(:, Ncx2)) - angle(X(:, Ncx1)) - d\varphi_0$$

nous devons également s'assurer que la phase varie toujours entre – pi et + pi

$$d\varphi = d\varphi - 2\pi . round(\frac{d\varphi}{2\pi})$$

et, finalement, la mise à jour de la phase *phi* :

$$\varphi = \varphi + d\varphi + d\varphi_0$$

3- La synthèse →TFCT inverse

On utilise la TFCT inverse pour revenir dans le domaine temporel en combinant les trames fenêtrées et en appliquant une TF inverse.
La fonction doit donc renvoyer un vecteur qui est le « son audio » dans le domaine temporel et interpolé selon le vecteur temps (t) choisi.