ESIEE PARIS

UNITE : OBL -TRAITEMENT DU SIGNAL

Projet : Vocodeur de phase

Traitement du signal audio

Modification de pitch et robotisation de la voix

Table des matières

OBJECTIFS DU PROJET ET ATTENDU DU RAPPORT

Objectif

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

Les principales différences par rapport aux séances de TP

1. **Création de votre propre programme**
   En plus de comprendre les « scripts » fournis, vous devez créer votre propre programme réalisant notamment la fonction principale du vocodeur de phase.
2. **Justification scientifique**
   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.
3. **Rédaction d'un rapport complet**
   À lui seul, il devrait permettre au lecteur de comprendre :
   - L'objectif de votre travail
   - Les explications et justifications théoriques
   - Votre démarche pour la réalisation du vocodeur de phase
   - Le principe des algorithmes utilisés (toutes les fonctions)
   - Les résultats obtenus commentés

Information importante

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 et 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 le Drive

1 TRAVAIL À RÉALISER

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...).

Les fonctions principales à réaliser sont :

1. **Le principe du vocodeur de phase** → interpolation dans le domaine fréquentiel
2. **La robotisation de la voix**

2 RENDU DU TRAVAIL

Information importante

Vous travaillerez en **binôme**. Chaque binôme déposera son rapport sur Google Drive dans l'espace prévu pour cela :

« `Projet` » → « `Rapports` »

en respectant son groupe. Bien entendu, vous déposerez 1 seul rapport par binôme et nommé **Projet_Groupe_Nom1_Nom2.zip**.

Contenu du fichier à rendre

Le fichier devra contenir :

- Tous les programmes Matlab (fichiers `.m`) nécessaires
- Les fichiers audio (`.wav` ou autre format dont vous disposerez) utilisés
- Le rapport sous format `.pdf`

Remarque

Attention : Pas de script Matlab dans le rapport (nous aurons les programmes).

Date limite : *La date limite pour deposer votre travail est le jeudi 27/11/2025.*

Tout rapport rendu après cette date se verra fortement penalisé par un retrait automatique de 5 points.

3 ORGANISATION GÉNÉRALE DU PROGRAMME

3.1 Introduction

Le vocodeur consiste à « coder » de la voix. Le vocodeur de phase agira essentiellement sur la phase du signal.

Remarque

Attention, ici le terme de « codage » n'est pas pris au sens du codage de données mais au sens « **modification** ».

3.2 Trois types de modifications de la voix

1. **Modification de la vitesse sans modifier le pitch**

   Le pitch étant lié à la fréquence fondamentale de la voix correspondant à la hauteur de la voix (grave, aiguë...).

   On garde la « même voix » mais les mots seront prononcés plus lentement ou plus rapidement.
2. **Modification du pitch sans modifier la vitesse**

   La voix devient plus aiguë ou plus grave, mais la vitesse de prononciation reste identique.
3. **Robotisation de la voix**

   Application d'un effet de synthèse vocale robotique.

3.3 Structure du programme principal

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 `PVoc.m`
2. Modification du pitch → appel de `PVoc.m` + ré-échantillonnage
3. Robotisation de la voix → appel de `Rob.m`

4 MODIFICATION DU PITCH ET DE LA VITESSE

4.1 Fonction `PVoc.m`

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

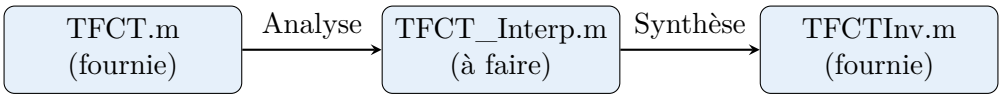
4.1.1 Principe de la TFCT

Information importante

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 ferons 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..

4.1.2 Architecture fonctionnelle



Ces 3 fonctions seront appelées dans l'ordre à partir d'une fonction plus générale `PVoc.m` fournie.

4.1.3 Différence entre modification de vitesse et de pitch

Différence clé

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

4.2 Travail à réaliser

Objectif

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)
- L'implémenter correctement

Remarque

Expérimentation : Lorsque vous modifiez le pitch, vous pouvez également écouter (et observer) ce que cela donne en additionnant le résultat au signal d'origine...
Exemples d'applications créatives :

- Faire des duos, des trios (test avec `Extrait.wav`)
- Ajouter des « sopranos », « contre-ténors » ou « basses » dans une chorale (test avec `Halleluia.wav`)

5 ROBOTISATION DE LA VOIX

5.1 Principe

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.

Information importante

Cet effet est également largement utilisé en musique depuis quelques décennies et 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.

5.2 Méthode simplifiée (à 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 projet).

Nous allons moduler 1 sinusoïde (ou plus exactement une exponentielle complexe) à une fréquence f_c :

$$\boxed{y_{rob}(t) = y(t) \cdot e^{j2\pi f_c t}} \tag{1}$$

Comme le résultat est un signal complexe, nous devrons récupérer la **partie réelle** de ce dernier.

5.3 Tests à réaliser

Expérimentations

La valeur de f_c déterminera le « degré de robotisation ». Vous pourrez faire les tests avec :

Valeur de f_c (Hz)	Qualité sonore
2000	Instructif mais peu judicieux
1000	Instructif mais peu judicieux
500	Intéressant
200	Bon résultat

Consigne : Expliquez ce qu'il se passe pour chaque valeur de f_c.
Les choix de f_c trop grands (2000 et 1000) ne sont pas judicieux en termes de son mais vous permettront de comprendre et d'expliquer ce qu'il se passe, notamment dans le domaine fréquentiel...

Remarque

Suggestions :

- La robotisation est plus « judicieuse » sur un signal de parole plutôt que sur une voix chantée (tests sur `Diner.wav`)
- Testez sur votre propre voix ! Il suffit d'enregistrer un extrait et d'appliquer la robotisation
- Proposez une autre valeur de f_c qui vous semblerait plus judicieuse

ANNEXE

A - PRINCIPE GÉNÉRAL DU VOCODEUR DE PHASE

Objectif

L'objectif est de « déplacer » le « pitch » d'un signal audio. Ce dernier étant défini 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

On considère une sinusoïde de fréquence f_0 (courbe du haut). La transposition à une fréquence 2 fois plus élevée, $2 \times f_0$, donnerait la courbe du bas dont la durée est 2 fois plus courte (le son serait alors 2 fois plus « rapide »).

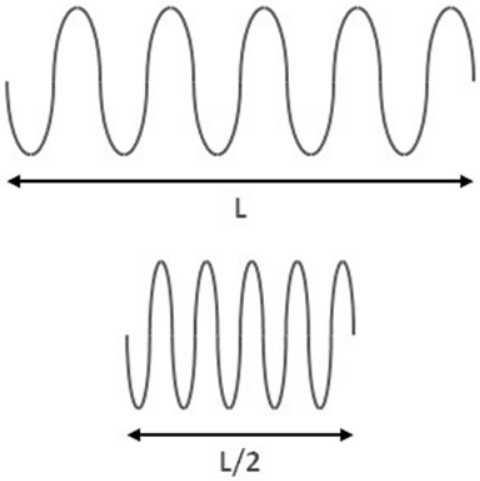


FIGURE 1 – Sinusoïde originale de longueur L et sinusoïde transposée de longueur L/2 (*Ref. http ://www.guitarpitchshifter.com/algorithm.html#33*)

Pour avoir l'impression d'avoir le même son mais prononcé (parole) ou joué (instrument) par une voix ou un instrument d'une octave au-dessus, il faudrait :

1. **D'abord interpoler** pour avoir une durée 2 fois plus grande (voir la Figure 2)
2. **Puis transposer** la fréquence

C'est ce que doit faire la fonction `TFCT_Interp.m` que vous devez créer.

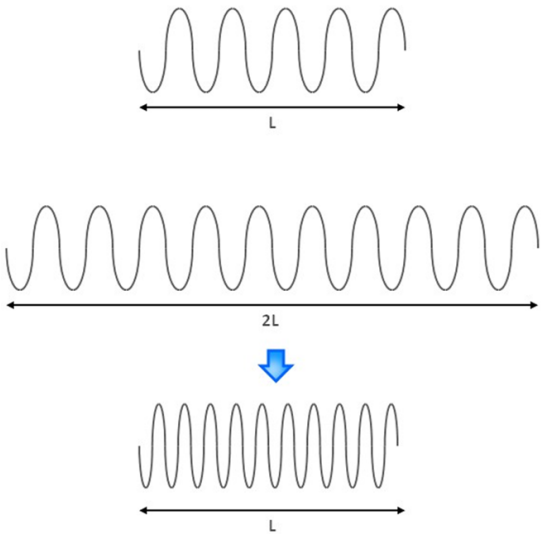


FIGURE 2 – Sinusoïde originale de longueur L et sinusoïde transposée de longueur L grâce à l'interpolation à 2L.

Transformée de Fourier à Court Terme (TFCT)

Information importante

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.
Solution : 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** ».

Trois étapes nécessaires dans le vocodeur de phase :

1. **Analyse** : Le passage dans le domaine fréquentiel par TFCT
   → fonction Matlab fournie : `TFCT.m`
2. **Traitement** : La transposition des trames
   → fonction Matlab **à coder** : `TFCT_Interp.m`
3. **Synthèse** : Le retour dans le domaine temporel
   → fonction Matlab fournie : `TFCTInv.m`

Schéma général

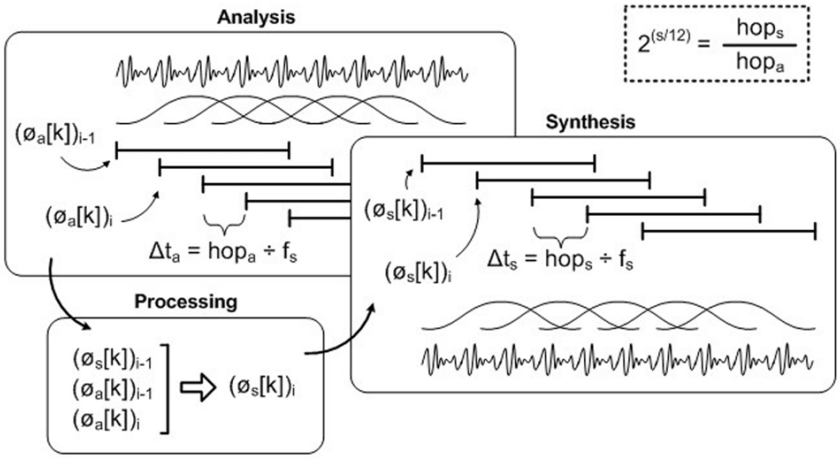


FIGURE 3 – Principe général du vocodeur de phase (*Ref. http ://www.guitarpitchshifter.com/algorithm.html#33*).

1. L'analyse par la 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 par une **matrice** avec :

- **N lignes** : nombre de points de la TF pour chaque trame
- **Ntr colonnes** : nombre de trames

Chaque colonne représente la TF de chaque trame

On ne garde que les échantillons aux fréquences positives de la TF (de 0 à $F_e/2$) :

$$N = 1 + \frac{N_{fft}}{2}$$

où N_{fft} est le nombre de points de calcul de la TF.

2. Le traitement par 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)
% X est la matrice issue de la TFCT
% t est le vecteur des indices sur lesquels doit être faite
% l'interpolation
% Nov est le nombre d'échantillons correspondant au chevauchement des
% fenêtres (trames) lors de la TFCT

[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);
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

</> Structure de l'algorithme

Chaque trame (colonne) de Y (numérotée par `Ncy`) est centrée à un instant t_n du vecteur temps (`t`).
Pour chaque valeur de `t` notée t_n, 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.**

Structure de la boucle :

1. **Début de la boucle :** pour chaque valeur t_n de t
   a) On récupère les 2 colonnes « adéquates » de X
      ⇒ Définir les indices `Ncx1` et `Ncx2` des 2 colonnes de X en fonction de la position de l'instant t_n
   b) On calcule le module `My` interpolé
   c) On calcule la colonne de Y :
      `Y(:, Ncy) = My .* exp(j*phi)`
   d) On met à jour la nouvelle phase `phi` qui sera utilisée pour le calcul de la prochaine colonne de Y
   e) On incrémente l'indice colonne de Y :
      `Ncy = Ncy + 1`
2. **Fin de la boucle**

Formules mathématiques

• **Calcul du module interpolé :**

$$M_y = \alpha|X(:, Ncx1)| + \beta|X(:, Ncx2)| \tag{2}$$

avec :

$$\alpha = 1 - \beta \tag{3}$$
$$\beta = t_n - \lfloor t_n \rfloor \tag{4}$$

$\lfloor t_n \rfloor$ est le symbole qui arrondi t_n à l'entier inférieur le plus proche (voir *floor* dans Matlab.

• **Mise à jour de la phase :**
Étape 1 : Calcul de la variation de phase entre les 2 colonnes successives de X

$$d\varphi = \text{angle}(X(:, Ncx2)) - \text{angle}(X(:, Ncx1)) - d\varphi_0 \tag{5}$$

Étape 2 : S'assurer que la phase varie toujours entre $-\pi$ et $+\pi$

$$\boxed{d\varphi \leftarrow d\varphi - 2\pi \cdot \text{round}\left(\frac{d\varphi}{2\pi}\right)} \tag{6}$$

Étape 3 : Mise à jour finale de la phase

$$\boxed{\varphi \leftarrow \varphi + d\varphi + d\varphi_0} \tag{7}$$

3. La synthèse par la 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.

Bon courage dans votre projet !