|  |  |
| --- | --- |
| LAPERTOT Raphaël  LAFOURCADE Anthony | IGS-3005  21/11/2018 |

Rapport  
Projet « Mariotiseur »



# Introduction

Ce projet « Mariotiseur » a été réalisé par Raphaël LAPERTOT et Anthony LAFOURCADE durant les mois de janvier et février 2019, dans le cadre de l’unité IGS-3005 « Traitement de signal » tenue par Nadia MADAOUI à l’ESIEE Paris.

Le but de ce projet est de réaliser sur MATLAB un vocodeur de phase offrant, sur un signal audio et mono donné, les fonctionnalités suivantes :

* Modification de la vitesse du signal (sans modification de la hauteur)
* Modification de la hauteur du signal (sans modification de la vitesse)
* Robotisation du signal
* « Mariotisation » du signal

*Mariotisation ?! Qu’est-ce que c’est que ça ?*

C’est le cœur de notre projet. Il s’agit d’une fonctionnalité permettant de faire les premières notes de musique du thème de Mario Bros. en utilisant un signal donné et en modifiant son pitch, sa hauteur, sa fréquence.

Ce rapport explique le fonctionnement de ces différentes fonctionnalités.

Les fichiers TFCT.m et TFCTInv.m sont déjà fourni et complets. Nous expliquerons également leur utilité dans le cadre de ce projet.

# **Modification de la vitesse**

Dans cette partie, on va essayer de réaliser une fonction permettant de ralentir un signal, ou au contraire l’accélérer, tout en gardant le même pitch, la même fréquence, la même tonalité.

Cette fonctionnalité est très pratique, et est très largement utilisée notamment dans les pubs où chaque seconde coûte des milliers : ils accélèrent souvent les dialogues pour gagner du temps tout en gardant la compréhensibilité de la voix.

## Récupération des fréquences du signal au cours du temps (TFCT.m)

Pour cette partie, un signal nous est donné en entrée, ainsi que sa fréquence d’échantillonnage, et le rapport d’accélération voulu (par exemple, 2 si on veut obtenir un signal deux fois plus rapide).

Une première idée pourrait être de modifier la fréquence d’échantillonnage lors de la lecture du signal donné. Cette idée fonctionne et permet d’avoir un signal accéléré ou ralenti, mais la hauteur du son s’en retrouve également affecté.

Nous allons devoir donc passer dans le domaine fréquentiel, puis essayer de recréer un signal temporel de longueur différente tout en gardant les mêmes fréquences que le signal donné.

Pour passer dans le domaine fréquentiel, une simple transformée de Fourier n’est pas pertinente, car le signal donné est aléatoire et n’est pas stationnaire. Cependant, on peut considérer des sous-signaux de courte durée de ce signal aléatoire comme stationnaire, du fait que pendant cette courte période, la hauteur du son ne varie que très peu.

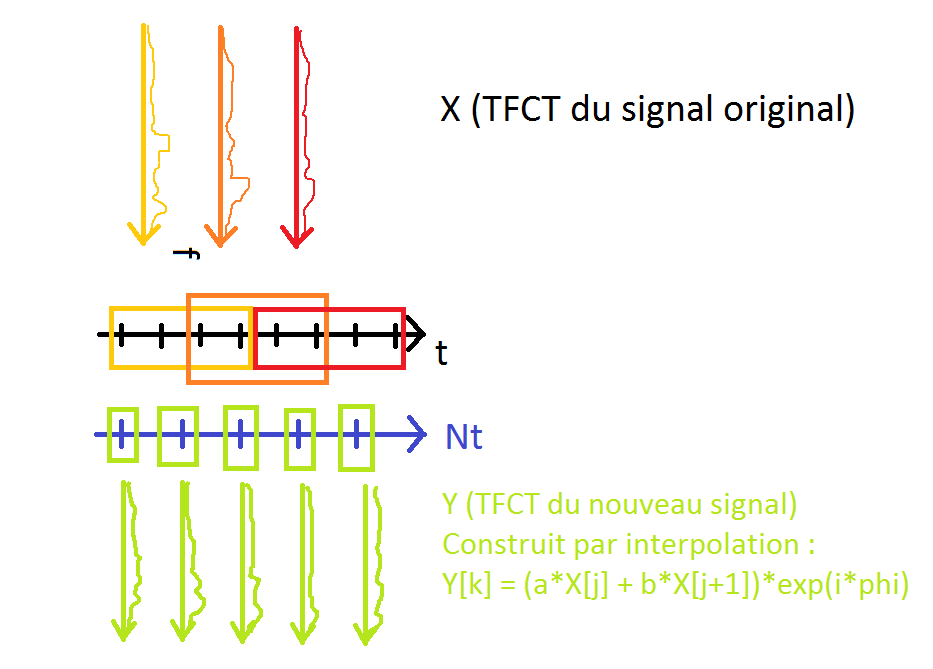
Nous allons donc découper notre signal en plusieurs sous-signaux de 1024 échantillons (correspondant à environ 80ms), se chevauchant et espacés temporellement d’une valeur appelée le « hop », qu’on a choisi ici arbitrairement à 1024/4 = 256 échantillons (~=20ms), puis faire à chacun une transformée de Fourier. Le fichier TFCT.m (pour Transformée de Fourier à Court Terme) permet de réaliser cette manipulation, et nous retourne le tableau des transformées de Fourier.

## Création du nouveau signal fréquentiel (TFCT\_Interp.m)

Maintenant que nous connaissons l’évolution des fréquences de notre signal au cours du temps, nous allons créer un nouveau tableau de transformée de Fourier, correspondant à la TFCT de notre nouveau signal.

Pour cela, on crée d’abord un nouveau vecteur de temps Nt modifié selon le rapport d’accélération voulu (par exemple, si le rapport vaut 2, ce vecteur vaut [0, 2, 4, 6, …]), et on essaye de créer les TFCT « étendues » pour le nouveau signal, par interpolation. Pour cela, on considère chaque valeur temporelle de notre nouveau signal temporel comme ayant subi une TF en cette valeur temporelle.

Le schéma suivant permet d’illustrer comment est formé ce nouveau vecteur de transformées de Fourier :



Avec : a = partie décimale de la valeur temporelle courante de Nt (encadrée en vert)

b = 1 – a

On se doit également de conserver la phase lors de la construction de ce nouveau vecteur de TF pour reproduire un son fidèle. C’est le rôle de l’exponentielle complexe, dont phi est recalculé à chaque étape.

De cette manière, le module de notre nouvelle transformée de Fourier est une « moyenne à poids » de deux transformées de Fourier de signal original (celles qui ont été faites pour les t encadrant la valeur temporelle courante de Nt), et déphasé de manière à avoir une phase proche de celles des TF originales, à chaque étape.

## Création du signal temporel à partir de la TFCT créée

Maintenant que nous avons la TFCT de notre nouveau signal, il nous suffit de faire sa TFCT inverse pour retrouver le nouveau signal temporel. Pour cela, on utilise la fonction TFCTInv.m fournie, avec les mêmes paramètres que lors de la réalisation de la TFCT du signal original.

# **Modification de la hauteur**

On cherche maintenant à faire le contraire : changer le pitch d’un signal sans modifier sa vitesse, sa durée.

Comme on peut désormais modifier la vitesse d’un signal sans modifier sa hauteur, il est très simple de faire le contraire. En effet, pour modifier la hauteur d’un signal sans modifier sa vitesse, il suffit de :

* Modifier la vitesse du signal
* Rééchantillonner ce nouveau signal de façon à retrouver la vitesse originale

Dans les faits, on utilise la fonction précédente pour créer un nouveau signal, puis on rééchantillonne ce nouveau signal grâce à la fonction resample de Matlab, en utilisant le même coefficient pour les deux fonctions.

Le rééchantillonnage permet de retrouver la vitesse initiale, tout en changeant le pitch. On obtient au final un signal de même longueur, mais d’un pitch différent selon le rapport donné.

Remarque : en sommant un signal à lui-même réhaussé, on obtient un effet rendant les voix monstrueuses, comme des voix d’aliens. Cet effet peut être très intéressant pour un tas d’autres projets !

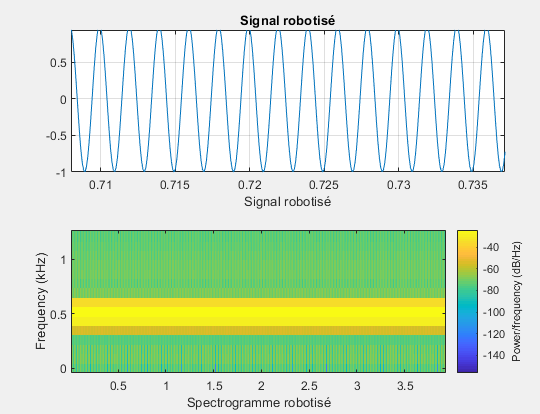
# **Robotisation**

Pour ajouter un effet robotique au signal, on joue sur sa phase en rajoutant une fréquence de robotisation Fc, indicatrice du degré de robotisation.

Dans les faits, on devrait passer dans le domaine complexe pour agir directement sur la phase, mais comme réaliser cette manipulation est hors-programme, nous allons nous contenter de rajouter une phase complexe au signal temporel, puis prendre sa partie réelle (un signal temporel est réel).

En déphasant le signal temporel avec cette exponentielle complexe à Fc (sans oublier de prendre la partie réelle uniquement pour retrouver un signal réel), on rajoute dans le domaine fréquentiel une fréquence (Fc) qui va venir perturber le signal temporel, donnant cette impression de robotisation.

Pour illustrer cet effet, on peut prendre un signal constant (y = ones(N,1)) et lui appliquer l’effet. Voici ce qu’on obtient :



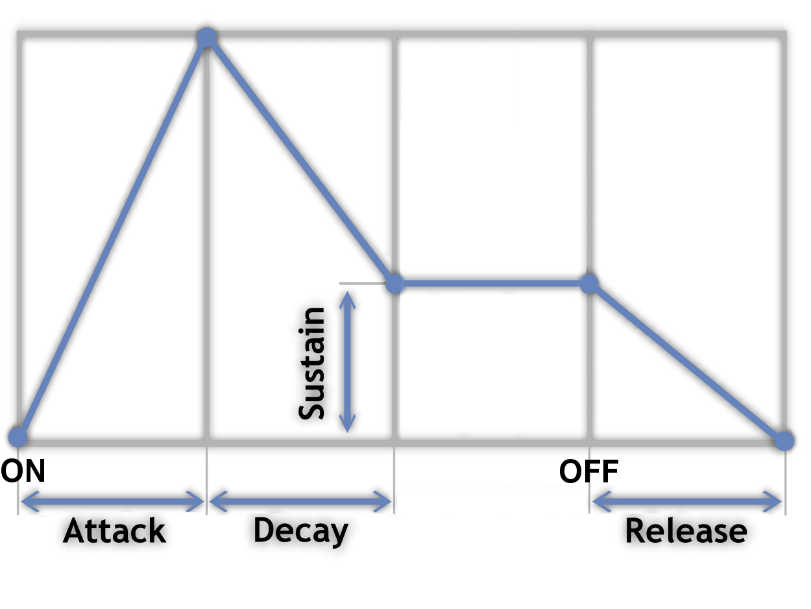
Augmenter Fc au-dessus de 1000Hz n’est pas très pertinent car on souhaite la plupart du temps robotiser une voix, et les voix se trouvent à des fréquences autour de 200~400Hz. De ce fait, elles ne seront pas très perturbées, et ça sera d’autres fréquences qui seront affectées.

C’est pour cela que nous conseillons d’utiliser une fréquence de robotisation Fc autour de 500Hz.

# **Mariotisation**

Nous arrivons enfin au cœur de notre projet. Le but de la « mariotisation », comme nous l’appelons, est de, à partir d’un signal donné, jouer les premières notes du thème principal du jeu Mario Bros.

Pour cela, il nous faut d’abord découper le signal donné pour n’en garder **qu’une partie qui nous servira de note** par la suite. Nous avons décidé de prendre un extrait de 1/3 de secondes au moment où l’amplitude du signal est maximale. Plus précisément, on commence notre extrait à 1/12ème de secondes avant le moment tmax où l’amplitude du signal est maximale, et on le termine après 1/3 de secondes, soit 3/12ème = ¼ de secondes après le même moment tmax. De cette manière, on obtient un extrait où l’amplitude max est au quart de sa durée, ce qu’on a trouvé optimal car souvent l’amplitude maximale est à la fin d’une attaque (attack) : il est donc important de garder toute l’attaque, mais également la chute (decay), la phase d’entretien et l’extinction (release).



Maintenant qu’on a notre extrait, il nous faut **l’augmenter** ou le **diminuer** pour lui faire faire les notes du thème de Mario Bros. On a donc besoin des notes de cette musique.

Le début est comme suit (exposant indiquant l’octave) : mi0 mi0 – mi0 – do0 mi0 – sol0 - - - sol-1 - - -

On a donc besoin de la liste des fréquences associées aux notes Do, Mi et Sol. Ces informations sont facilement trouvables sur Internet (source : <http://jeanjacques.dialo.free.fr/frequenc.htm> )

Maintenant, il nous faut déterminer la **fréquence** à plus haute énergie de **notre extrait**, pour pouvoir ensuite **l’augmenter** ou **le diminuer** **du bon coefficient**. Pour cela, rien de tel qu’une transformée de Fourier. On réalise donc une transformée de Fourier de notre extrait, puis on y récupère la fréquence dont l’énergie est maximale.

Il nous faut désormais choisir à quelle octave jouer le thème de Mario Bros. Nous avons décidé de prendre l’octave de la note de l’extrait, donc de jouer des notes proches de la note de l’extrait original.

Ensuite, nous nous attaquons à la modification du pitch. Pour cette étape, nous disposons de la fonction de modification du pitch détaillée auparavant. Il nous suffit de l’utiliser en utilisant comme coefficient :

Il nous faut faire cette étape pour chaque note : Do0, Mi0, Sol0 et Sol-1.

Enfin, une fois ces nouveaux extraits calculés, on dispose des notes à jouer : il nous reste plus qu’à créer un signal et à les y mettre dans l’ordre.

Si on prend un BPM de 90 (=3/2 battements par secondes), on a besoin de 4 battements (4 noires) pour y mettre toutes les notes. On va donc créer un nouveau signal d’une longueur de :

Ensuite, les notes s’enchainent en double-croche, qui dure :

Les notes de Mario Bros. étant mi0 mi0 – mi0 – do0 mi0 – sol0 - - - sol-1 - - -, on construit donc notre signal en mettant :

* l’extrait en Mi0 au début (1)
* un autre extrait en Mi0 à 1/6 \* Fs
* un autre Mi0 à 1/6 \* 3 \* Fs
* un Do0 à 1/6 \* 5 \* Fs
* un Mi0 à 1/6 \* 6 \* Fs
* un Sol0 à 1/6 \* 8 \* Fs
* enfin, un Sol-1 à 1/6 \* 12 \* Fs

Et voilà, on a notre signal mariotisé ! Dans le programme Vocodeur.m, je vous propose de mariotiser un extrait où un chihuahua aboie : le rendu est particulièrement efficace.