LAPERTOT Raphaël
LAFOURCADE Anthony

IGS-3005
12/12/2018

Compte rendu du TP2
Introduction au filtrage numérique

Partie A

A1 :

En traçant le spectre du signal audio, un pic d'énergie apparaît à environ 82Hz: il s'agit de l'harmonique de la contrebasse, qu'on cherche à retirer. On a donc Fcb = 82 Hz.

Afin de supprimer cette fréquence, nous allons utiliser un filtre.

Le filtre est défini par une fonction de transfert H(z), qu'on décompose en un numérateur N(z) et un dénominateur D(z).

Comme on veut annuler des fréquences, c'est à dire les mettre à 0, on cherche à annuler le numérateur, et donc avoir des zéros, pour la fréquence Fcb.

Les zéros sont deux racines complexes conjuguées (car on veut annuler qu'une seule fréquence) de la forme $re^{+-j(theta)}$. Il nous faut donc déterminer theta et r.

On choisit r = 1 car on veut annuler, mettre à 0, donc le module vaut forcément 1.

Pour theta, on fait un produit en croix :

0 ➔ 0

2pi ➔ Fe

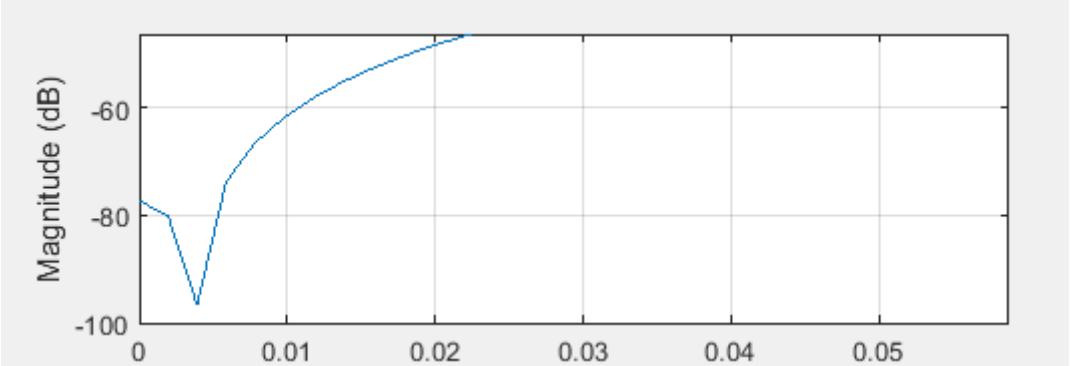
theta ➔ Fcb

On en déduit que theta = 2*pi*Fcb/Fe

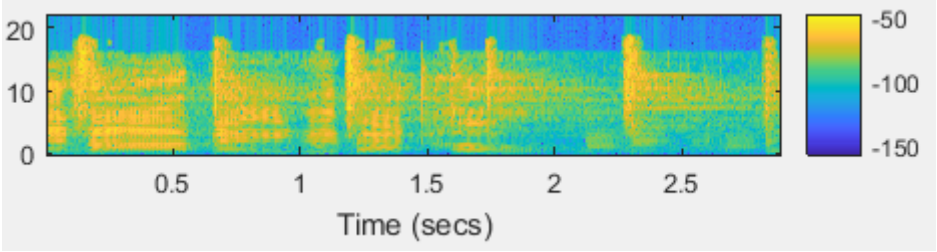
Matlab nous calcule les coefficients du polynôme associé à ces racines grâce à la fonction poly(racines).

Pour le dénominateur, on choisit D(z) = 1 pour éviter de créer des instabilités. On crée du coup un filtre FIR.

Notre filtre fait bien un pic d'amplitude vers le bas, aux alentours de 82Hz.



Sur le spectrogramme, on observe des énergies faibles pour les fréquences basses :



Enfin, à l'écoute, la contrebasse a disparu, mais le son est généralement moins fort également.

A2 :

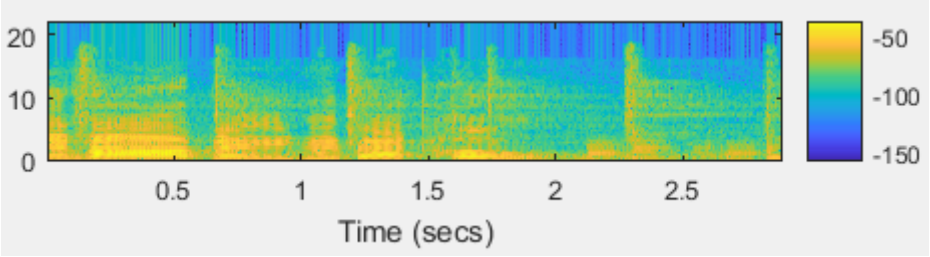
On trouve Fnat grâce au spectre : on trouve un pic de fréquence entre 200 et 400 hz, à 329Hz. On choisit donc Fnat = 329Hz.

On choisit N(z) = 1 car on veut amplifier, pas atténuer. On crée donc un filtre IIR.

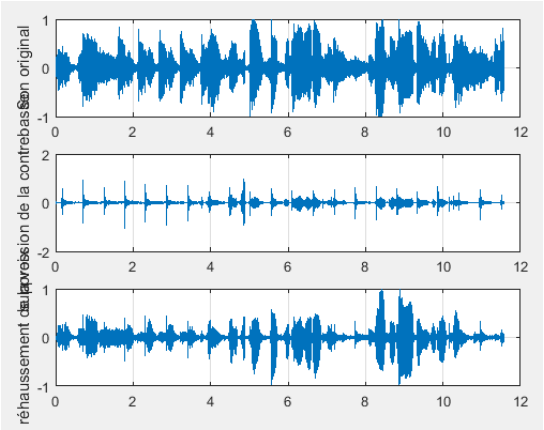
Pour les pôles, ils seront également de la forme $re^{+-j(theta)}$. On ne peut pas prendre r=1 car on ne peut pas diviser par 0 : on va donc s'en rapprocher en prennant r=0.9.

On calcule theta de la même manière qu'en A1, et on a donc theta = 2*pi*Fnat/Fe.

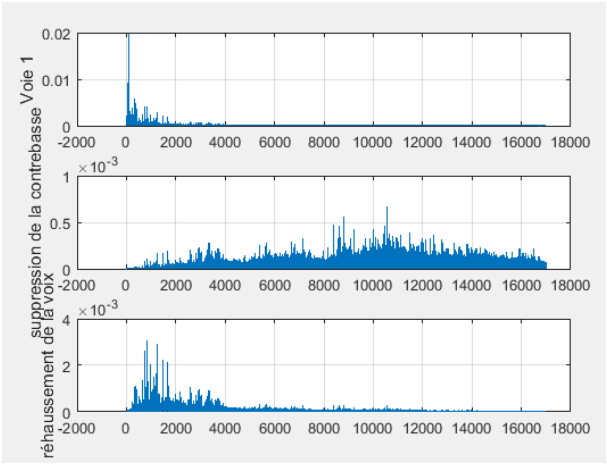
On observe bien une amplification aux alentours de Fnat pour la fonction de transfert, et le spectrogramme montre des énergies plus élevées autour de Fnat :



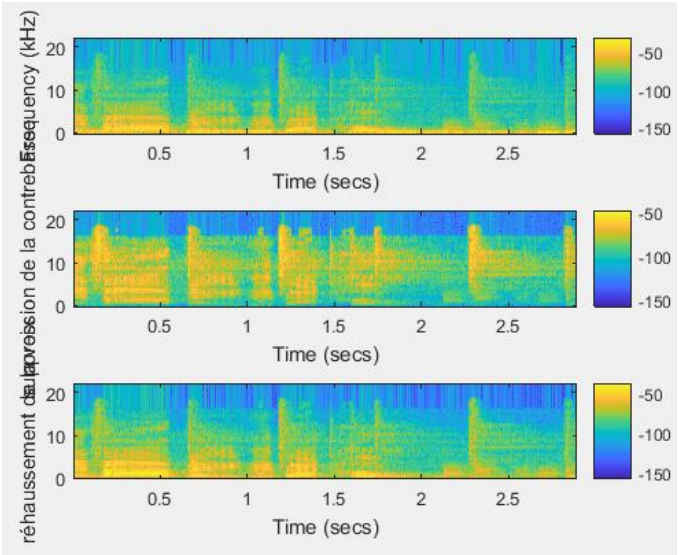
En comparant les courbes des signaux temporels, on voit bien que le 1er filtre atténue le son, et que le 2eme filtre amplifie le son, particulièrement dans la deuxième moitié, ce qui fait sens car c'est dans la deuxième moitié que Natalie chante.



En comparant les courbes des spectres, on voit bien que le 1er filtre atténue les graves, et que le 2e filtre amplifie les fréquences autour de 329Hz comme voulu.



Les différents spectrogrammes montrent bien que le 1er filtre a retiré de l'énergie aux fréquences basses (ligne bleue en bas sur le 2e spectrogramme), et que le 2e filtre a rehaussé l'énergie dans les fréquences moyennes (ligne jaune plus vif autour de 300Hz sur le 3e spectrogramme).



Partie B :

B1 :

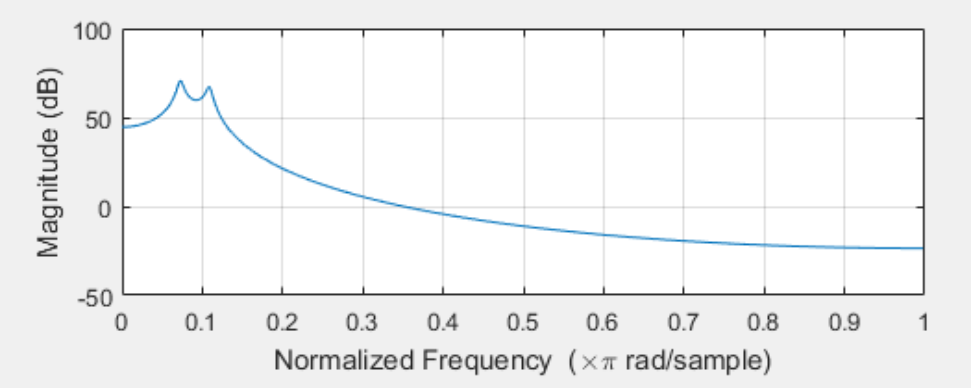
Comme nous voulons accentuer certaines fréquences, il nous faut des passe-bandes ; on va utiliser un filtre IIR, d'ordre 2 car le gain d'un filtre d'ordre 1 est monotone, alors que le gain d'un filtre d'ordre 2 aura un « pic » vers le haut autour d'une fréquence.

B2 :

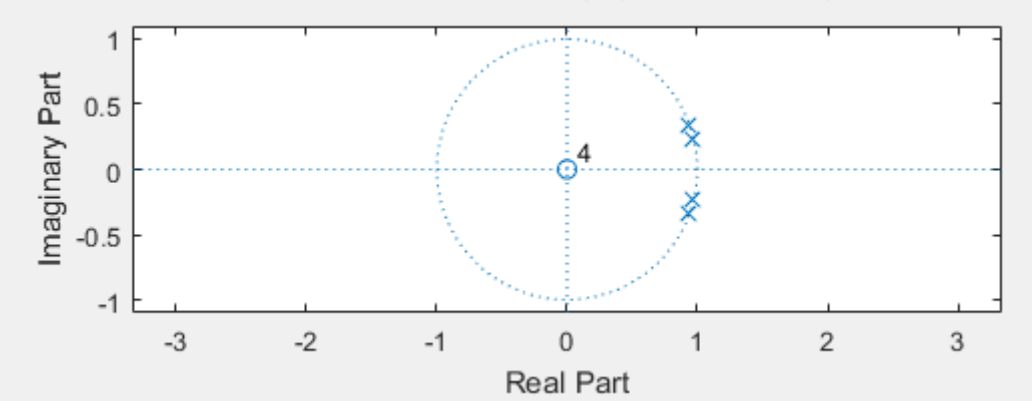
Pour créer un son simulant une voyelle, le programme procède en plusieurs étapes :

- On déclare toutes les valeurs de départ (fréquence d'échantillonnage, fréquences des voyelles à accentuer, et fréquences des voix)
- On fixe la personne qui parle (homme) et la voyelle qu'il va prononcer (un « a » ici)
- On crée le signal d'entrée du filtre (qui vise à simuler les vibrations des cordes vocales par l'air). C'est un peigne de Dirac. Pour le créer, on crée d'abord un vecteur nul de N échantillons, puis on détermine le pas du peigne : c'est $F_e/F_0 \Leftrightarrow T_0/T_e \Leftrightarrow N_0$, puis on complète le signal nul précédemment créé avec des diracs (mise à 1) tous les N_0.
- On détermine les pôles, puis les coefficients du dénominateur. Comme on veut amplifier deux fréquences, on va avoir 2x2 pôles conjugués à déterminer : ça donne un filtre d'ordre 4. Comme on veut amplifier, on veut diviser par quelque chose de proche de 0, tout en restant stable ; on prend donc r ~= 1 et r<1 ➔ r = 0.99.
  Pour theta, on réalise le même produit en croix que d'habitude pour faire correspondre un angle à la fréquence voulue : $theta_{1,2} = Fa_{1,2} * 2pi / Fe$
  On détermine les coefficients des polynômes grâce à la fonction poly de Matlab vue précédemment.
  Enfin, on réalise le filtrage grâce à la fonction filter de Matlab, qui nous donne notre signal de sortie.

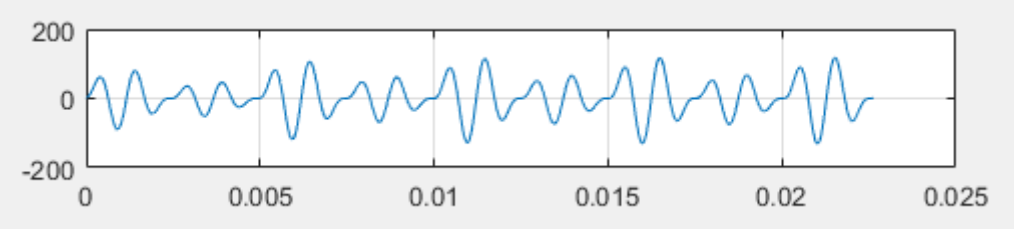
Sur le gain du filtre, on observe bien deux pics correspondant aux deux fréquences qu'on a voulu accentuer :



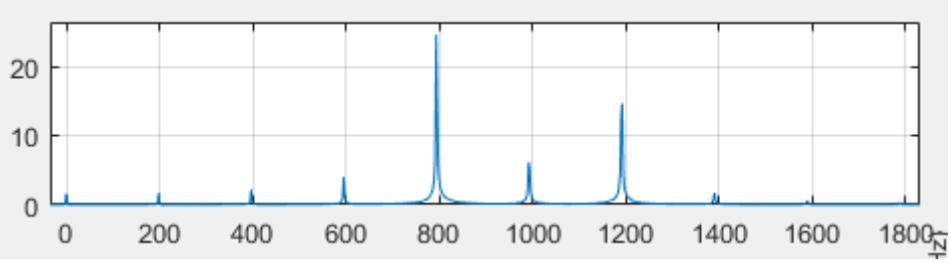
On observe les 4 pôles qu'on a créés, correspondant aux fréquences qu'on a voulu accentuer, sur le plan complexe :



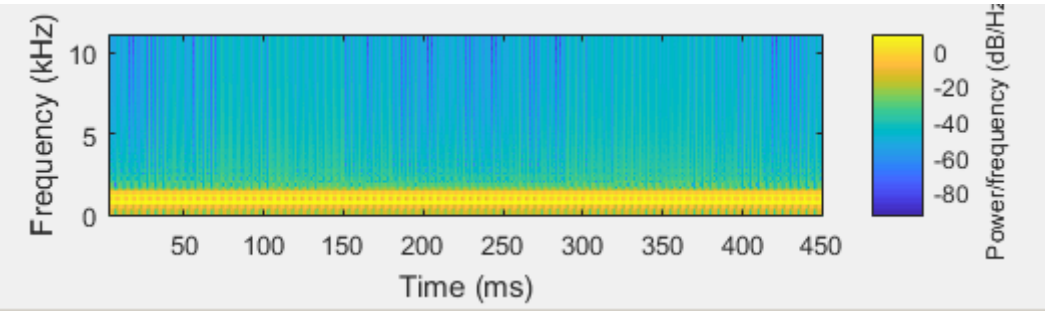
La figure temporelle montre une somme de sinusoïdes. Il n'y a pas grand-chose à dire dessus, si ce n'est qu'on observe une pseudo-période tous les 5ms :



La figure fréquentielle montre des pics de fréquences en 800 et 1200Hz : ce sont les fréquences de la voyelle « a », qu'on a décidé d'accentuer, et on retrouve le peigne de diracs tous les 200Hz qu'on a créé, correspondant à la fréquence de l'homme :



Le spectrogramme montre une énergie constante au cours du temps, ce qui fait sens car on fait la même voyelle tous le long. On constate des énergies plus élevées au niveau des fréquences qu'on a accentué, ce qui rejoins l'observation de la figure fréquentielle.



B3 :

Les voyelles créées paraissent encore synthétiques. Pour améliorer et rendre ces sons plus naturels, on pourrait :

- Augmenter le nombre d'harmoniques, ce qui rendra le son plus proche du son de la voyelle
- Affiner F0, choisissant une valeur plus juste, ce qui rendra un son plus proche de l'humain.
- Ajouter une enveloppe pour imiter un son avec une attack, un sustain et un release plus proche de l'humain

B4 :

Pour augmenter le nombre d'harmoniques, il suffit de modifier les vecteurs des fréquences des voyelles fa, fe, … en ajoutant une troisième fréquence, qu'on trouve dans l'annexe. On obtient des sons qui ressemblent déjà un peu plus à des sons de voyelles.

En faisant quelques recherches, on trouve que la fréquence moyenne chez les hommes vaut 125Hz, et 210Hz pour une femme, contre 300Hz pour des enfants. En remplaçant F_Homme et F_Femme par ces valeurs, on obtient un son plus proche d'une voix humaine.

Pour ajouter une enveloppe, on utilise la fonction interp1 pour créer une enveloppe à partir d'un vecteur temps, un vecteur avec des valeurs temporelles spécifiques et un vecteur de valeurs prises à ces valeurs temporelles. Interp1 va donc créer l'enveloppe : on a ensuite juste à multiplier le signal de sortie par cette enveloppe. En faisant plusieurs tests d'enveloppe, on arrive à des « prononciations » plus proche de ce que sortirai de la bouche d'un être humain.