TP Synthèse de parole

Roland Badeau - Gaël Richard

Dans ce TP, vous allez implémenter des fonctions d'analyse/synthèse du signal de parole basées sur le système PSOLA. Ces fonctions seront testées sur des signaux que vous pourrez trouver sur le site pédagogique de l'UE SI340.

Ces signaux sont échantillonnés à la fréquence `Fs` = 22050Hz. Vous pourrez les charger sous Matlab en tapant par exemple `load aeiou` ; ils seront alors stockés dans la variable `s`. Pour les écouter, vous pourrez taper `soundsc(s,Fs)`.

1 Extraction des marques d'analyse

Dans un premier temps, vous programmerez la fonction

```
function A = AnalysisPitchMarks(s,Fs)
```

qui extrait les marques d'analyse. Les arguments `s` et `Fs` sont respectivement le signal à analyser et la fréquence d'échantillonnage. La matrice `A` renvoyée contiendra les instants et les pitchs associés à chaque marque d'analyse. Plus précisément, `A` sera constituée de trois lignes, telles que `A(1,n)` $= t_a(n)$ est l'instant associé à la n^{eme} marque d'analyse, `A(2,n)` $=$ voise(n) est un booléen qui indique si le signal est voisé ou non au voisinage de cette marque, et `A(3,n)` $= P_a(n)$ est le pitch associé à cette même marque (c'est à dire la période en nombre d'échantillons) dans le cas voisé, ou vaut 10ms dans le cas non voisé.

Vous aurez en particulier besoin d'un détecteur de pitch. Pour gagner du temps, vous pourrez utiliser la fonction `periode.m`, dont le code Matlab vous est fourni avec les signaux d'exemple. Cette fonction requiert deux arguments : un signal à court terme `x` extrait de `s`, et la fréquence d'échantillonnage `Fs` (les autres arguments sont facultatifs), et renvoie un couple `[P, voise]` ou `voise` est un booléen qui indique si `x` est voisé ou non, et `P` est la période en nombre d'échantillons dans le cas voisé, ou vaut 10ms dans le cas non voisé.

Précisons maintenant comment déterminer les marques d'analyse. Par souci de simplicité, nous ne chercherons pas à aligner la marque $t_a(n)$ sur le début d'une onde glottique. Pour calculer $P_a(n)$ et $t_a(n)$, on procédera par récurrence sur $n \geq 1$:

- extraction d'une séquence `x` qui commence à l'instant $t_a(n-1)$, et dont la durée est égale à $2.5\,P_a(n-1)$;
- calcul de $P_a(n)$ et voise(n) à l'aide de la fonction `periode` ;
- calcul de $t_a(n) = t_a(n-1) + P_a(n)$.

L'algorithme sera initialisé en posant $t_a(0) = 1$ et $P_a(0) = 10$ms.

2 Synthèse et modification des échelles temporelle et fréquentielle

Pour effectuer la synthèse du signal, nous devons commencer par définir des marques de synthèse. Celles-ci seront stockées dans une matrice `B` constituée de deux lignes, telles que `B(1,k)` $= t_s(k)$ est l'instant associé à la k^{eme} marque de synthèse, et `B(2,k)` $= n(k)$ est le numéro de la marque d'analyse associée à cette même marque de synthèse. Pour commencer, vous pourrez effectuer une synthèse sans modification, en posant :

- `B(1, :)` $=$ `A(1, :)` ;
- `B(2, :)` $= [1, 2, 3, \ldots]$.

2.1 Synthèse du signal

Vous allez maintenant programmer la fonction

```
function y = Synthesis(s,Fs,A,B)
```

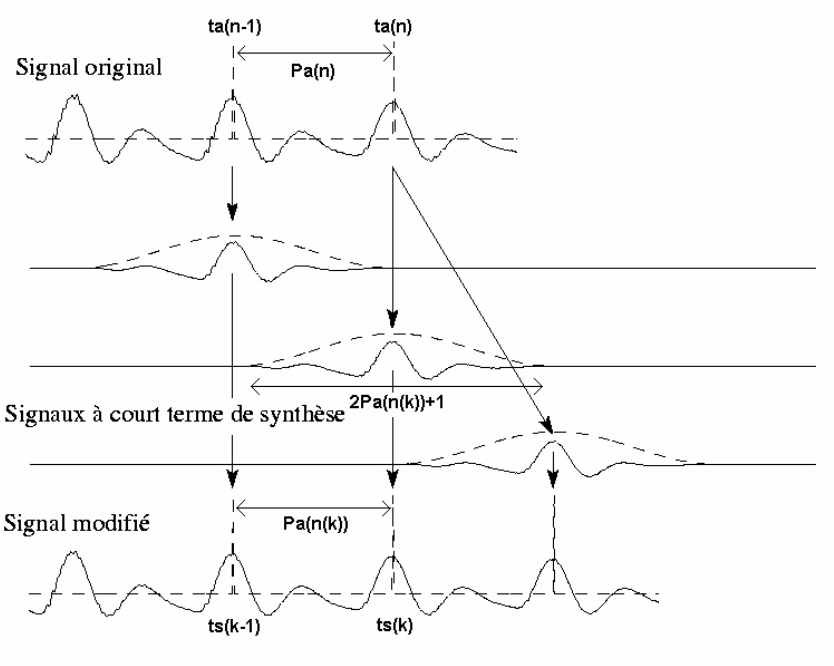

FIG. 1 – Modification de l'échelle temporelle

qui calcule le signal de synthèse `y` à partir du signal original `s`, de la fréquence d'échantillonnage `Fs`, des marques d'analyse stockées dans la matrice `A` et des marques de synthèse stockées dans la matrice `B`. La synthèse s'effectue très simplement par récurrence sur $k \geq 1$ (le vecteur `y` étant préalablement initialisé au vecteur nul de dimension $t_s(k_{\text{end}}) + P_a(n(k_{\text{end}}))$) :

- extraction d'une séquence `x` centrée en $t_a(n(k))$ et de longueur $2\,P_a(n(k)) + 1$;
- fenêtrage de `x` par une fenêtre de Hanning (utiliser la fonction `hann`) ;
- addition-recouvrement de la séquence `x` fenêtrée sur $y(t_s(k) - P_a(n(k)) : t_s(k) + P_a(n(k)))$.

2.2 Modification de l'échelle temporelle

Il s'agit maintenant de déterminer des marques de synthèse qui vont modifier l'échelle temporelle d'un facteur α, autrement dit de déterminer une matrice `B` telle que la durée du signal synthétisé par la fonction `Synthesis` soit égale à celle du signal original `s` multipliée par α. Cette opération sera effectuée par la fonction

```
function B = ChangeTimeScale(alpha,A,Fs)
```

qui calcule la matrice `B` à partir du facteur α, des marques d'analyse stockées dans `A`, et de la fréquence d'échantillonnage `Fs`. On peut procéder par récurrence sur $k \geq 1$, en utilisant un indice $n(k)$ non entier :

- $t_s(k) = t_s(k-1) + P_a(\lfloor n(k) \rfloor)$;
- $n(k+1) = n(k) + \frac{1}{\alpha}$.

L'algorithme sera initialisé en posant $t_s(0) = 1$ et $n(1) = 1$. On prêtera attention à ne stocker que des valeurs entières dans la matrice `B`.

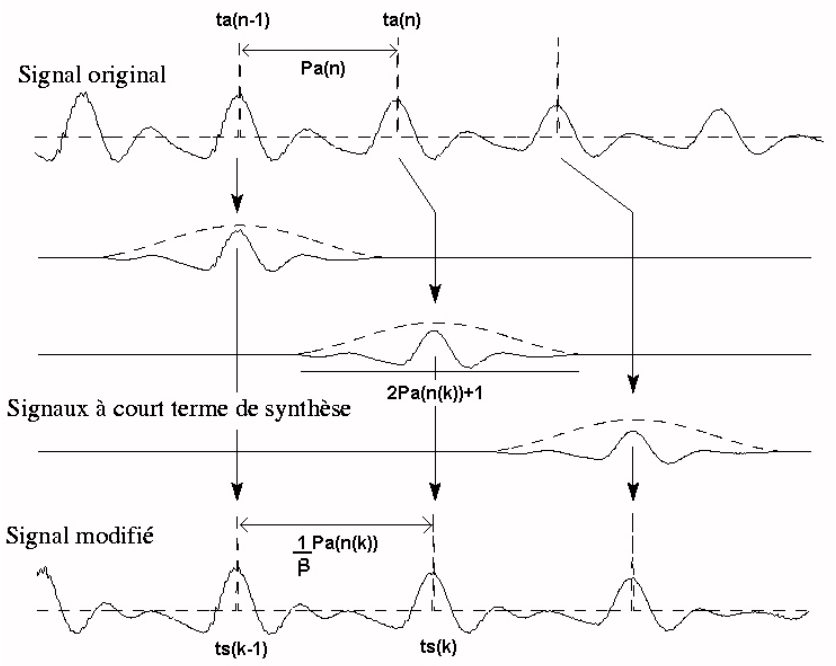

FIG. 2 – Modification de l'échelle fréquentielle

2.3 Modification de l'échelle fréquentielle

Vous allez maintenant effectuer l'opération duale de la précédente, c'est à dire déterminer des marques de synthèse qui vont modifier l'échelle fréquentielle d'un facteur β, autrement dit déterminer une matrice `B` telle que la fréquence fondamentale du signal synthétisé par la fonction `Synthesis` soit égale à celle du signal original `s` multipliée par β. Cette opération sera effectuée par la fonction

```
function B = ChangePitchScale(beta,A,Fs)
```

qui calcule la matrice `B` à partir du facteur β, des marques d'analyse stockées dans `A`, et de la fréquence d'échantillonnage `Fs`. On pourra, comme dans le cas précédent, procéder par récurrence sur $k \geq 1$, en utilisant un indice $n(k)$ non entier *et* des instants de synthèse $t_s(k)$ non entiers, et en faisant bien attention de distinguer les cas voisé et non voisé :

- si la marque d'analyse d'indice $\lfloor n(k) \rfloor$ est voisée, $\text{scale}(k) = \frac{1}{\beta}$, sinon $\text{scale}(k) = 1$;
- $t_s(k) = t_s(k-1) + \text{scale}(k) \times P_a(\lfloor n(k) \rfloor)$;
- $n(k+1) = n(k) + \text{scale}(k)$.

Là encore, on prêtera attention à ne stocker que des valeurs entières dans la matrice `B`.

2.4 Modification conjointe de l'échelle temporelle et de l'échelle fréquentielle

Pour finir, vous allez programmer une fonction qui modifie conjointement les deux échelles :

```
function B = ChangeBothScales(alpha,beta,A,Fs)
```

où les arguments sont définis comme précédemment. Le corps de la fonction sera quasiment identique à celui de `ChangePitchScale` ; il vous suffira donc de modifier celui-ci de manière adéquate.