M1 IPCC : Bases du Traitement du Signal

TP : Synthèse de filtres RIF

A la fin de ce TP, vous devez rendre un compte-rendu sur une copie identifiée par votre login de manière visible (pas sous le rabat). N'oubliez pas de sauvegarder les fichiers .sce modifiés, qui seront récupérés de manière automatique à la déconnexion, de même que les figures indiquées.

Il est rappelé qu'un compte-rendu de TP doit être rédigé avec soin et lisiblement. Toute observation doit faire l'objet à la fois d'une description concise et d'une interprétation.

1 Synthèse par fenêtrage et utilisation d'un filtre passe-bas

1) Montrer qu'un filtre passe-bas idéal de fréquence de coupure normalisée $f_c = 1/4$ a pour répoonse impulsionnelle :

$$h(n) = \frac{1}{2}\mathrm{sinc}\Big(\frac{n\pi}{2}\Big)$$

2) Ouvrez *TP_1.sce* dans l'éditeur de scilab. Ce programme génère $L = 2M + 1$ échantillons de $h(n)$ (pour $n \in [-M; +M]$), calcule la réponse fréquentielle H par une FFT sur $NFFT = 512$ points et affiche le module de H.

Lancez le programme. Expliquer l'aspect de la courbe obtenue.

Calculez et dessinez le gabarit théorique de la réponse fréquentielle. Décommentez l'appel de la fonction `trace_gabarit` et donnez-lui les bons paramètres pour vérifier votre résultat. Voir l'annexe pour la conversion entre les dB et les valeurs linéaires.

3) On souhaite une atténuation supérieure à 50 dB dans la bande atténuée et la même largeur de bande de transition que précédemment. Comment s'y prendre ? Testez votre proposition en sauvegardant le fichier sous un autre nom (*TP_1bis.sce*). Pour changer les extrema des axes de la figure, voir l'annexe. Enregistrez votre figure sous *TP_1.eps*. Que constatez-vous ? Pour quelle valeur de N la réponse fréquentielle rentre-t-elle dans le gabarit ?

4) Soit $x(n) = cos(2\pi f_1 n) + cos(2\pi f_2 n)$, avec $f_1 = 1/8$ et $f_2 = 3/8$. Dessiner le spectre d'amplitude théorique de x, en dB. Soit y la réponse du filtre synthétisé à la question 2, lorsque l'entrée est x. Dessiner le spectre d'amplitude théorique de y, en dB.

Ouvrez *TP_2.sce*. Ce programme génère $N = 64$ échantillons de x et affiche le spectre de cette séquence calculé par TFD sur 512 points. Il calcule y par convolution et affiche le spectre d'amplitude de y sur la même figure que celui de x.

Lancez *TP_2.sce*. D'après les paramètres de l'analyse spectrale (fenêtrage rectangulaire de longueur N), expliquer pourquoi le résidu de la deuxième sinusoïde subsistant après filtrage n'est pas observable.

2 Synthèse par échantillonnage : étude d'un débruiteur

La réduction de bruit pour les signaux audio est une problématique largement étudiée dans le domaine des communications, notamment en téléphonie mobile, où le bruit de fond de l'environnement dégrade la qualité de la transmission. La difficulté est que l'on ne dispose généralement d'aucune information sur le bruit ni sur le signal originel.

La parole n'étant pas stationnaire, le traitement est réalisé trame par trame, selon le schéma de la figure 1. Une analyse spectrale est effectuée sur des fenêtres de 32 ms (256 échantillons) se recouvrant à 50%. Pour chaque trame analysée, on construit un filtre débruiteur spécifique, qui permet de traiter la deuxième moitié de la trame analysée (la première moitié étant traitée par le filtre précédent, grâce au recouvrement de 50%). On construit ainsi le signal débruité par blocs de 128 échantillons.

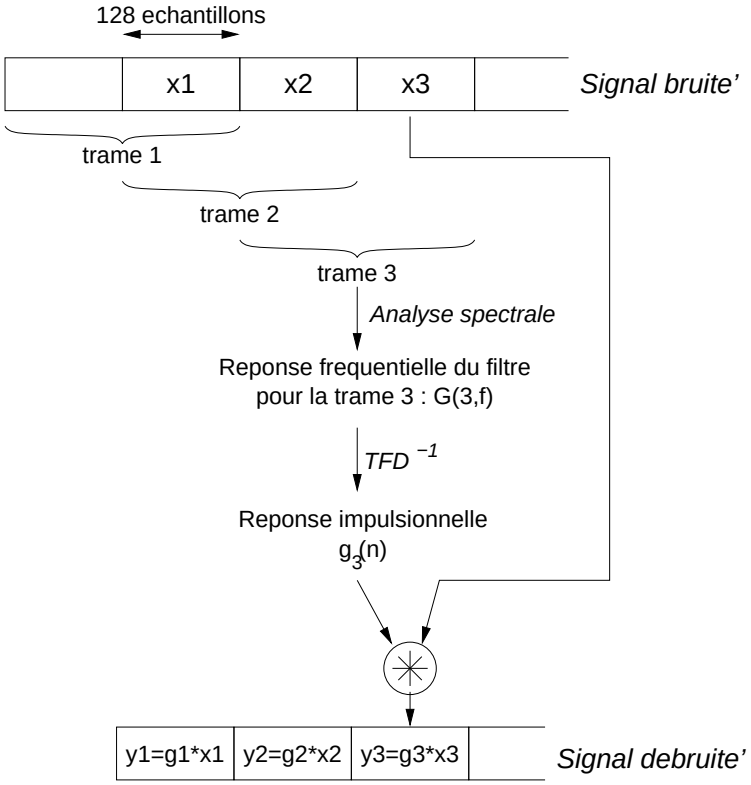

FIG. 1 – Traitement par bloc d'un signal de parole.

Si l'on note $s(n)$ le signal original et $b(n)$ le bruit, le signal bruité s'exprime :

$$x(n) = s(n) + b(n)$$

Le débruitage est réalisé selon les principes de l'atténuation spectrale à court terme. A chaque trame p, dans le domaine fréquentiel,

$$X(p, f) = S(p, f) + B(p, f)$$

où $S(p, f)$, $B(p, f)$ et $X(p, f)$ représentent respectivement le spectre de la p^{eme} trame de signal original, de bruit et de signal bruité. L'objectif est de trouver un gain spectral $G(p, f)$ tel que $S(p, f) \simeq G(p, f)X(p, f)$. La p^{eme} trame de signal bruité x est alors filtrée par le filtre de réponse impulsionnelle $g(n) = \mathrm{TFTD}^{-1}[G(p, f)]$.

On utilise un filtre de Wiener [1], qui minimise l'erreur quadratique moyenne, définie par :

$$EQM = \mathrm{E}\Big[\Big(S(p,f) - G(p,f)X(p,f)\Big)^2\Big]$$

Ce filtre s'exprime ainsi :

$$G(p,f) = \frac{\overline{|S(p,f)|^2}}{\overline{|S(p,f)|^2} + \overline{|B(p,f)|^2}} = \frac{RSB_{prio}(p,f)}{1 + RSB_{prio}(p,f)} \tag{1}$$

où RSB_{prio} désigne le Rapport Signal à Bruit *a priori*, défini par :

$$RSB_{prio}(p,f) = \frac{\overline{|S(p,f)|^2}}{\overline{|B(p,f)|^2}}$$

La difficulté est d'estimer ces grandeurs.

La grandeur $\overline{|B(p,f)|^2}$, appelée densité spectrale de puissance du bruit et notée $\gamma_b(p,f)$, est estimée ici pendant les périodes d'inactivité vocale, par un lissage récursif du spectre d'amplitude du signal :

$$\gamma_b(p,f) = \begin{cases} \lambda\gamma_b(p-1,f) + (1-\lambda)|X(p,f)|^2 & \text{si la trame } p \text{ contient uniquement du bruit} \\ \gamma_b(p-1,f) & \text{sinon} \end{cases}$$

où $\lambda \in [0;1]$ est le facteur de lissage, le lissage étant d'autant plus fort que λ est proche de 1.

Le RSB_{prio} est estimé selon la méthode d'Ephraïm et Malah [2], qui repose à la fois sur l'estimation $\hat{S}(p-1,f)$ de la trame précédente et sur une estimation locale de RSB_{prio} :

$$RSB_{prio}(p,f) = \beta\frac{|\hat{S}(p-1,f)|^2}{\gamma_b(p,f)} + (1-\beta)\mathrm{Max}\Big(RSB_{inst}(p,f), 0\Big)$$

où β est proche de 1 et RSB_{inst} désigne le RSB instantané, défini comme une estimée locale de RSB_{prio} :

$$RSB_{post} = \frac{|X(p,f)|^2 - \gamma_b(p,f)}{\gamma_b(p,f)}$$

Au cours des manipulations qui suivent, ne fermez pas les fenêtres graphiques qui s'ouvrent.

1) Lancer *demo_deb.sce*. L'exécution est terminée lorsque "fini" apparaît dans scilab. Ce programme ouvre le fichier son *F1_ori.wav*, ajoute du bruit et débruite. Pour cela, il segmente d'abord le signal entre la parole et le silence grâce à la fonction *dav.sci* (Détection d'Activité Vocale, normalement réalisée parallèlement au débruitage). Puis le signal est débruité *via* la fonction *debruite.sci*. Les signaux bruité et débruité sont enregistrés respectivement dans *F1_b.wav* et *F1_deb1.wav*.

Ecouter les fichiers son : original, bruité et débruité. Que remarquez-vous ?

Le programme affiche les réponses impulsionnelle g et fréquentielle (en amplitude) $|G(f)|$ du filtre pour la trame 63. La réponse fréquentielle est représentée uniquement pour les fréquences positives. A partir de la réponse impulsionnelle, expliquez le phénomène que vous avez perçu dans le signal débruité.

2) On montre que si l'on permute la première et la deuxième moitié de la réponse impulsionnelle d'un filtre RIF, le module de la réponse fréquentielle reste inchangé (voir annexe). Comme c'est le module qui nous intéresse ici, expliquer comment cette permutation permettrait d'éviter le phénomène précédent. Décommentez la ligne 67 de *debruite.sci*, qui effectue la permutation. Relancez *demo_deb.sce* en fixant $opt = 2$ au début du programme.

On rappelle l'expression de la convolution qui a lieu lors d'un filtrage RIF :

$$y(n) = \sum_{i=0}^{L-1} g(i)x(n-i)$$

avec L la longueur de la réponse impulsionnelle g. D'après la réponse impulsionnelle obtenue, pourquoi peut-on dire que la sortie est retardée de 128 échantillons par rapport à l'entrée du filtre ?

Le signal débruité est enregistré sous *F1_deb2.wav*. Ecoutez et comparez aux autres fichiers son.

3) Observez comment est réalisée la convolution dans *debruite.sci*. Combien de multiplications-accumulations sont-elles réalisées par échantillon ? Pour limiter la complexité du filtrage et le retard, ainsi que pour rendre le filtre moins "agressif" (*i.e.* débruiter sans trop distordre le signal), on souhaite limiter la réponse impulsionnelle à 63 coefficients. La synthèse du filtre dans la partie précédente était déjà réalisée par défaut par échantillonnage fréquentiel, puisque la réponse fréquentielle est calculée à partir des TFD des signaux. Pour réduire la taille de la réponse impulsionnelle, on peut soit procéder à un nouvel échantillonnage fréquentiel sur 64 points, soit fenêtrer la réponse impulsionnelle précédemment obtenue.

a) Dans *debruite.sci*, décommentez et complétez les lignes 70 à 72. Il faut :
- construire une réponse fréquentielle G_{ech} sur 64 points en prélevant un échantillon sur quatre de G (voir annexe) ;
- calculer la réponse impulsionnelle g_{ech} par TFD inverse, permuter les deux moitiés puis éliminer le premier coefficient, de manière à avoir une réponse symétrique de longueur 63 ;
- calculer la réponse fréquentielle correspondante G_{ech256} sur 256 points par TFD avec zéro-padding (voir *TP_1.sce*).

Relancez *demo_deb.sce* en fixant $opt = 3$. Le signal débruité est enregistré sous *F1_deb3.wav*. Sur la même figure que $|G|$ (fig. 0), $|G_{ech256}|$ s'affiche en bleu et $|G_{ech}|$ avec un rond noir par échantillon fréquentiel. Enregistrez votre figure sous le nom *deb_3.eps*.

La courbe de G_{ech256} correspond à la transformation fréquentielle qui va réellement s'appliquer au spectre du signal. Quel est l'inconvénient de cette méthode d'échantillonnage fréquentiel ?

Quel est le retard introduit par ce filtre ? (voir fig. 2)

b) Dans *debruite.sci*, décommentez et complétez les lignes 75 et 76. Il faut :
- Construire une réponse impulsionnelle tronquée g_{fen} en appliquant à g une fenêtre de Hamming de longueur 63 centrée sur le maximum.
- Calculer la réponse fréquentielle correspondante G_{fen} sur 256 points par TFD avec zéro-padding.

Relancez *demo_deb.sce* en fixant $opt = 4$. Le signal débruité est enregistré sous *F1_deb4.wav*. Sur la même figure que $|G|$ (fig. 0), $|G_{fen}|$ s'affiche en rouge.

Comparez $|G_{fen}|$ à $|G|$ et à $|G_{ech256}|$. Enregistrez votre figure sous *deb_4.eps*.

Ecoutez *F1_deb3.wav* et *F1_deb4.wav*, comparez-les entre eux et à *F1_deb2.wav*.

3 Annexe

3.1 Permutation circulaire de réponse impulsionnelle

Soit un filtre de réponse impulsionnelle $h(n)$ de durée finie N paire. On définit $h'(n)$ par :

$$h'(0...N-1) = [h(\frac{N}{2}...N-1), h(0...\frac{N}{2}-1)]$$

Alors

$$\begin{aligned}
H'(f) &= \sum_{n=0}^{N-1} h'(n)\mathrm{e}^{-\mathrm{j}2\pi fn} \\
&= \sum_{n=0}^{N/2-1} h(n+N/2)\mathrm{e}^{-\mathrm{j}2\pi fn} + \sum_{n=N/2}^{N-1} h(n-N/2)\mathrm{e}^{-\mathrm{j}2\pi fn} \\
&\quad \text{Changements de variables} \\
&= \sum_{n=N/2}^{N-1} h(n)\mathrm{e}^{-\mathrm{j}2\pi fn}\mathrm{e}^{j\pi N} + \sum_{n=0}^{N/2-1} h(n)\mathrm{e}^{-\mathrm{j}2\pi fn}\mathrm{e}^{-j\pi N} \\
&= (-1)^N \sum_{n=0}^{N-1} h(n)\mathrm{e}^{-\mathrm{j}2\pi fn} \\
&= (-1)^N H(f)
\end{aligned}$$

Ainsi, $|H'(f)| = |H(f)|$.

3.2 Manipulation des vecteurs

Attention, en scilab les indices des vecteurs commencent à 1 au lieu de 0.

Pour prélever 1 échantillon sur K du vecteur x, de l'indice n_1 à l'indice n_2 :

```
x_ech = x(n_1:K:n_2);
```

Pour permuter la première et la deuxième moitié d'un vecteur ligne x de longueur 10 :

```
 x = [x(6:10),x((1:5)];
```

Suppression des 2 premiers et des 2 derniers échantillons de x :

```
x = x(3:8);
```

Mulitplication d'un vecteur ligne y de longueur N par une fenêtre de Hamming :

```
y_fen = y.*window('hm',N);
```

3.3 Valeurs linéaires et valeurs logarithmiques

Soient $x > 0$ et y tels que $y = x_{\mathrm{dB}} = 20\log(x)$. Alors $x = 10^{\frac{x}{20}}$. En scilab, cela s'écrit :

```
y = 20*log10(x);
x = 10^(y/20);
```

3.4 Courbes avec scilab

`xset("window",0)`	Sélection ou création de la figure 0
`xbasc()`	Effacement du contenu de la figure
`plot2d(x,y,c)`	x = vecteur des abscisses, y = vecteur des ordonnées, c= couleur. Code des couleurs : 1 : noir -9 : 1 rond noir par échantillon 2 : bleu 3 : vert 4 : jaune 5 : rouge ...
`plot2d([x,y,c,"011"," "`, `[x_min y_min x_max y_max])`	Définition des bornes de la figure

Références

[1] P. Scalart et J. Vieira Filho, "Speech Enhancement Based on a a priori Signal to Noise Estimation", IEEE Intl. Conf. on Acoustic, Speech and Signal Processing, Atlanta, Etats-Unis, Vol. 2, pp. 629–632, Mai 1996.

[2] Y. Ephraïm et D. Malah, "Speech Enhancement Using a Minimum Mean Square Error Short-Time Spectral Estimator", IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-32, n° 6, pp 1109–1121, décembre 1984.