DIRO
IFT 3205

TRAVAIL PRATIQUE

Filtres RIF et RII

Max Mignotte

DIRO, Département d'Informatique et de Recherche Opérationnelle.
http : //www.iro.umontreal.ca/∼mignotte/ift3205
e-mail : mignotte@iro.umontreal.ca

1 Analyse d'un Filtre Inconnu

Ce TP consistera à étudier quelques applications d'analyse et de synthèse de quelques filtres RIF et RII en traitement du signal et de la parole. Dans un premier temps, on considère le filtre numérique défini par l'équation récurrente suivante

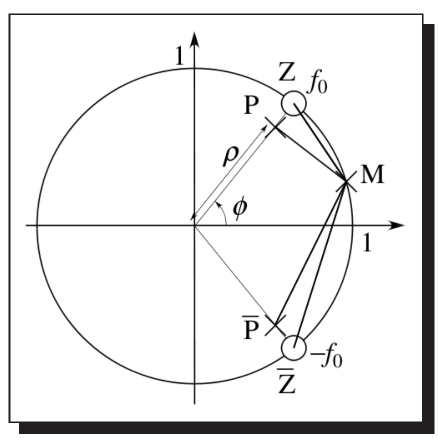
$$y(n) - 2\rho \cdot \cos\theta \cdot y(n-1) + \rho^2 \cdot y(n-2) = x(n) - x(n-1)$$

Dans le premier cas, on se donne $\rho = 0.99$ et $\theta = \pi/8$ et on prendra comme entrée à ce filtre un signal aléatoire dont l'amplitude est comprise entre 0 et 200.

1. Implémenter cette équation récurrente et générer le signal de sortie créé par ce filtre pour les 256 premiers échantillons. Visualiser les $y(n)$ obtenus.
2. Comparer le signal de sortie pour des valeurs de $\rho < 1$ du style $\rho = 0.5$ et pour une valeur $\rho = 1.05$? Puis pour $\rho = 0.99$ et $\theta = \pi/16$.

2 Suppression d'une Composante Sinusoidale

La rejection consiste à supprimer une composante fréquentielle de fréquence f_0 (que l'on considère parasite) dans un signal. Pour synthétiser un filtre numérique qui réalise un tel filtrage, l'idée la plus simple consiste à s'aider des propriétés pôles/zéros des filtres numériques vus en cours et donc de placer un zéro sur le cercle unité en $f = f_0$, i.e., à la fréquence numérique que l'on désire rejeter. Comme on veut une fonction de transfert réelle (et donc une réponse impulsionnelle réelle), il faut aussi placer sur ce cercle unité un autre zéro, conjugué du premier. De ce fait, le numérateur $N(z)$ a alors la forme



$$N(z) = \left(1 - \frac{\exp(2\,\pi j f_0)}{z}\right)\left(1 - \frac{\exp(-2\,\pi j f_0)}{z}\right)$$

Si on se limite à ce filtrage, le gain de ce filtre ne sera pas suffisant. C'est la raison pour laquelle on place un pôle à proximité de chaque zéro. Ici on prend $\rho \exp(2\pi j f_0)$ et $\rho \exp(-2\pi j f_0)$ avec ρ inférieur mais proche de 1 ($\rho = 0.99$). Ainsi lorsque $z = \exp(2\pi j f)$ se trouve "éloigné" des couples "pôles-zéro", le gain de ce filtre numérique sera construit pour être sensiblement égal à 1. En utilisant toutes ses considérations, cela conduit à la fonction de transfert suivante (avec $\theta = 2\pi f_0$)

$$H(z) = \frac{1 - 2\,\cos\theta\, z^{-1} + z^{-2}}{1 - 2\,\rho\cos\theta\, z^{-1} + \rho^2\, z^{-2}} \qquad (1)$$

1. Écouter puis convertir & lire le fichier son SOUNDFILE.WAV (i.e., créer le fichier SOUNDFILE.DAT avec le logiciel SOX et visualiser ensuite ce fichier par notre script VIEWSIG.SH).
2. Créer, par programme, un nouveau signal qui ajoute à ce dernier (i.e., SOUNDFILE.WAV) une fréquence sinusoìdale parasite de fréquence 500 Hz et d'amplitude 1.0. Écouter ou visualiser le nouveau signal sonore dégradé obtenu.
3. Implémenter et effectuer le filtrage de ce signal dégradé (que l'on appellera SOUNDFILE.DAT et SOUNDFILE.WAV) avec le filtre donné par la fonction de transfert donnée par l'Équation (1).

3 Égalisation

Idéalement, si un canal de transmission était parfait, un signal véhiculé et transmis ($y(t)$) par celui-ci serait identique au signal d'entrée ($x(t)$). Cela conduirait à une fonction de transfert du canal de transmission égale à 1, i.e., $C(z) = 1$ ou de façon équivalente à une réponse impulsionnelle associée à cette fonction de transfert égale à un Dirac ($c(t) = \delta(t)$) car on le rappelle

$$y(t) = x(t) = x(t) * \delta(t)$$

Dans le cas réel, un canal de transmission n'est pas parfait et les trajets multiples de propagation du signal véhiculé ($x_{\text{IN}}(t)$) dans celui-ci introduit une déformation du signal d'entrée et ses dégradations (linéaires) sont modélisées par la fonction de transfert du canal. Considérons dans cet exercice un modèle de canal dont la fonction de transfert est de la forme

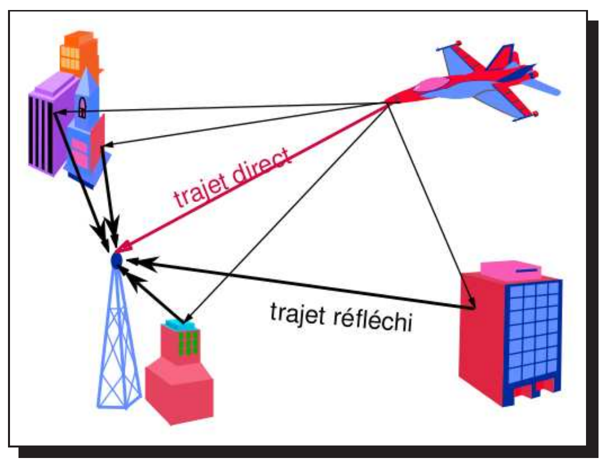


FIG. 1 – *Exemple de canal de tranmission avec des trajets multiples de propagation*

$$C(z) = (1 - 2z^{-1})(1 - \frac{1}{3}z^{-1})$$

1. Simuler numériquement (i.e., implémenter sur ordinateur) la sortie de ce canal (pour les premiers 10 échantillons) si on appliquait à celui-ci une impulsion de Dirac et, dans un deuxième temps, un échelon unité.
2. On considère comme signal d'entré à ce canal le signal SOUNDFILE.WAV. Simuler numériquement ce canal grâce à sa fonction de transfert $C(z)$ et créer, écouter, visualiser le signal d'entrée SOUNDFILE.WAV à la sortie de celui-ci (comparativement au signal d'entrée non dégradé SOUNDFILE.WAV) et calculer le RSB de ce son dégradé.

L'égalisation ou filtrage adapté est une opération consistant à palier le mieux possible les déformations apportées par un canal de transmission. Une technique d'égalisation possible est la technique dîte par “zero forcing” qui est la définition d'un filtre numérique qui tente d'inverser exactement la fonction de transfert du canal (ce qui est *a priori*, précisément le but recherché, idéalement par l'égalisation). Cependant on peut s'apercevoir que cette démarche souffre de deux défauts : d'abord, $C(z)$ peut posséder des zéros de module supérieur à un, ce qui induit des pôles instables si ce filtre est causal. C'est le cas de notre exemple si on considère $H(z) = 1/C(z)$ ce filtre inverse.

3. Calculer la réponse impulsionnelle de ce filtre inverse et discuter de sa stabilité.

On peut néanmoins contourner partiellement ce problème de stabilité en introduisant un retard lors de la résolution ce qui permet de prendre en compte une éventuelle partie non causale. On considère maintenant le filtre (de gain unité et) de fonction de transfert

$$P(z) = \frac{z-2}{1-2z}$$

4. Calculer $H(z)\,P(z)$ et implémenter numériquement à la sortie de ce canal la correction numérique modélisé par $H(z)\,P(z)$ et écouter de nouveau le signal d'entrée SoundFile.wav à la sortie de celui-ci après correction et calculer le RSB de ce son restoré.

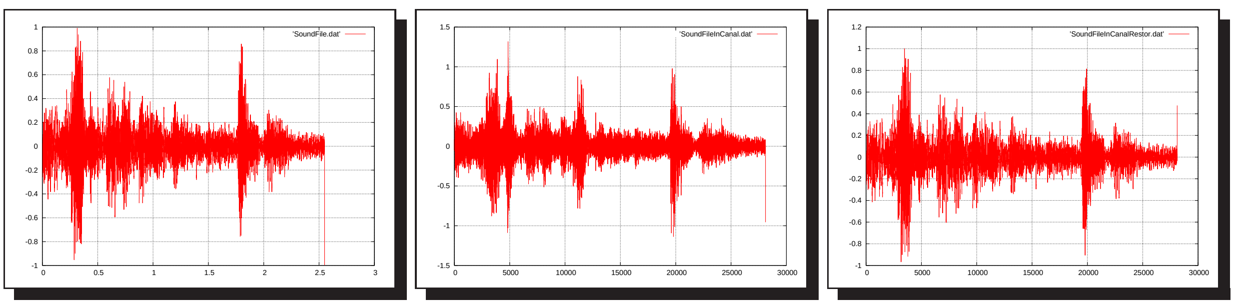


Fig. 2 – Le fichier son SoundFile.wav avant la dégradation (image de gauche) par le canal, puis dégradé (image du milieu) par le canal puis restoré (image de droite).

Remise & Rapport

Vous devez rendre électroniquement avec la commande *remise* le rapport et le(s) programme(s) fait en C (avant la date de remise spécifiée dans le fichier *barême*) dans le répertoire TP<Numéro du Tp>. Les programmes devront se compiler et s'executer sur Linux tel qu'indiqué dans le barême. Le rapport (si celui-ci est demandé dans le fichier barême) doit être concis et clair.

Logiciels Utilisés

Pour la conversion des signaux de parole/son (format Wave) nous disposons sous Linux d'un logiciel libre qui permet d'enregistrer, de convertir ou d'éditer du son. Ce logiciel est le programme sox.

• Pour convertir un fichier Wave dans un format texte plus simple (qui sera plus facile à enregistrer puis à traiter par votre programme C) contenant une colonne pour la variable temporelle (échantillonée régulièrement) et une colonne pour l'amplitude du signal (voir ci-contre), il faut taper sur la console la commande suivante

```
sox fichier.wav   fichier.dat
```

```
; Sample Rate 8000
; Channels 1
               0    0.00012207031
        0.000125    9.1552734e-05
         0.00025    3.0517578e-05
        0.000375    0.00024414063
          0.0005    0.00048828125
        0.000625    0.00048828125
         0.00075    0.00033569336
        0.000875    0.00030517578
           0.001    0.00033569336
        0.001125    0.00021362305
         0.00125    3.0517578e-05
        0.001375   -6.1035156e-05
          0.0015   -0.00018310547
        0.001625   -0.00030517578
         0.00175    -0.0002746582
        0.001875    -0.0002746582
           0.002   -0.00018310547
        0.002125   -3.0517578e-05
```

• Pour obtenir un fichier exploitable par nos programmes (visualisable par notre script ViewSig.sh et enregistrable dans un vecteur dans votre programme par la fonction LoadSignalDat("fichier",&length), on enlevera l'entête de ce fichier, i.e., les deux première ligne dont la première indique le taux d'échantillonnage que l'on retiendra. Une fois enregistrée dans un vecteur dans votre programme par la fonction LoadSignalDat("fichier",&length), et une fois traitée, puis enregistrée dans un fichier texte par la fonction SaveSignalDat("fichierTraité",VecteurSignal,length), on peut faire l'opération inverse, i.e., convertir ce fichier en un fichier au format Wave en ajoutant une première ligne (à ce fichier) indiquant le taux d'échantillonnage (; Sample Rate 8000 par exemple pour une période d'échantillonnage de $T = 1/8000$ seconde) et ensuite taper sur la console la commande suivante

```
sox fichier.dat   fichier.wav
```