TP Traitement du signal et compression

Ce travail pratique nécessite un casque audio, et le logiciel Scilab [1].

1 Écouter le repliement du spectre

Considérons le signal à temps continu $s(t) = \cos[\theta(t)]$ de durée τ que l'on veut échantillonner à la fréquence F_e. On appelle *fréquence instantanée* la fonction $F_i(t) = (1/2\pi)d\theta/dt$. Étudions par exemple le cas où la fréquence instantanée varie linéairement en fonction du temps : $F_i(t) = F_0 + \lambda t$. Cela signifie que la fréquence du signal part de la valeur F_0 et augmente linéairement avec le temps.

1. En supposant que $\theta(0) = 0$, quelle est la fonction $\theta(t)$?

   On prend $F_0 = 1000$ Hz, $\lambda = 1000$ Hz/s, $F_e = 8000$ Hz et $\tau = 2$ s.

2. Exprimez le pas de temps T_e en fonction de la fréquence d'échantillonage F_e.

   Quelle est la syntaxe en langage Scilab pour écrire le vecteur temps ?

3. La commande `sound(s,Fe)` permet d'écouter le son échantillonné contenu dans le vecteur `s`. Qu'entendez-vous lorsque $\lambda = 3000$ Hz/s ?

   Interprétez ce résultat dans le cadre de vos connaissances sur le théorème d'échantillonnage de Shannon.

   Confirmez votre écoute visuellement en dessinant un spectrogramme du signal (utilisant une transformée de Fourier à fenêtre glissante), en utilisant la fonction *Func_ specgram.sci* [2]. Quelle est la signification des paramètres d'entrée p et n ? Prendre $p = 256$ dans un premier temps, puis varier p et n pour améliorer la lisibilité de l'image.

2 Écouter le bruit de quantification

Le but de cet exercice est de quantifier la hauteur d'un son et d'écouter le résultat lorsque la quantification augmente. On se place dans le cas d'une quantification uniforme où l'amplitude crête à crête $2A$ du signal est découpée en L intervalles égaux.

1. Choisissez un son .wav quelconque dans le dossier *commun*. Grâce à la fonction `wavread`, donnez le nombre de bits par échantillon b de ce son. C'est la quantification originelle du signal.

1. http ://www.scilab.org/
2. implémentée page 220 du livre de M. Bergounioux, Mathématiques pour le traitement du signal, Dunod (2010)

2. Pour la suite, on normalise l'amplitude du signal A à 1 (elle varie donc de -1 à +1). Quelle est la syntaxe Scilab permettant d'effectuer cette normalisation ?

   Une manière de quantifier est d'utiliser une commande du type `y=ceil(y*N)/N` où $N = L/2$. Quelle est son défaut par rapport au codage informatique binaire ?

3. Le but est de partir d'une valeur très élevée de N (faible quantification), puis de la diminuer en écoutant le son qui en résulte. Quelle est la valeur de départ de N (sa valeur maximale) ?

   À partir de quelle valeur de N_{min} entendez-vous un bruit parasite ?

   Dans ce dernier cas, exprimez le rapport signal sur bruit $R_{sb} = 10log_{10}(\frac{I_s}{I_b})$ (en dB, formule vue en cours) en fonction de L. La syntaxe pour la fonction logarithme décimal est `log10`.

   Pourquoi le bruit de quantification est-il audible, sachant qu'un niveau sonore de 80 dB (conversation normale) masque les sons de 50 dB ?

   La quantification est maximale lorsque le signal ne peut prendre que deux valeurs (0 ou 1). Essayez : êtes-vous capables de percevoir la mélodie ?

4. Visualisez le bruit de quantification en fonction du temps : pour cela, il faut soustraire le signal quantifié au signal originel. Quel est le spectre de ce bruit ? S'approche-t-il plus d'un *bruit rose* ou d'un *bruit blanc* ? (ces notions ont été vues en cours).

5. Quel est le taux de compression obtenu avec N_{min} ?

3 Compression audio avec pertes

L'analyse par banc de filtres est à la base des algorithmes de compression audio tels que le MP3. On découpe le domaine fréquentiel en M sous-bandes afin d'appliquer une compression différente pour chaque sous-bande selon des modèles psychoacoustiques. Ici, nous allons considérer une technique brutale de compression : la suppression pure et simple de la contribution d'une sous-bande à l'aide d'un seuil énergétique. Ainsi si l'énergie dans une sous-bande est inférieure à un certain pourcentage, nous allouons 0 bits pour le codage de l'échantillon dans cette bande de fréquence.

1. Chargez le son *BBC.wav* et dessinez son chronogramme. Quel est le nombre N d'échantillons ? Quelle est sa durée τ ? Quelle est la fréquence d'échantillonnage F_e ?

2. Visualisez son spectre. Quel est le pas en fréquences Δf ?

3. Création d'un **banc de filtres** : l'intervalle fréquentiel entre 0 et Fe/2 est découpé en M sous-bandes (prendre M=8), qui sont indexées par la lettre k ($k \in \{1, ..., M\}$). Un filtre passe-bandes idéal peut être modélisé par une fonction porte de largeur fréquentielle $L = n\Delta f$. Son amplitude est contenue dans le vecteur : `filtre=[zeros(1,(k-1)*n) ones(1,n) zeros(1,N-(n+1)*(k-1))]`. Quelle est la largeur d'un seul filtre passe-bandes, en fonction de F_e et M ?

4. Pour chaque sous-bande, on multiplie les coefficients de Fourier complexes par le filtre correspondant. On range les valeurs dans une matrice (qu'on nomme `fd`) pour les manipuler plus facilement. Quelle est la taille de cette matrice ?

5. Calculer l'énergie E dans chaque sous-bande `E(k)=diag(y(k,:)*y(k,:)')/2` puis dessiner l'histogramme des énergies avec la commande `bar`. Donner la syntaxe pour exprimer E(k) en % de l'énergie totale, et le vecteur E qui en découle.

   On considère ici que l'énergie d'une bande est trop faible lorsqu'elle n'excède pas 5%. Quel est le nombre K de signaux restants ?

   Annuler les signaux filtrés à négliger `fd(K+1:M,:)=0`.

6. Grâce à la TF inverse, reconstruire le signal dans chaque sous-bande. La synthèse du signal total est obtenue en sommant tous les signaux avec `sum`. Ecoutez le son synthétisé. Quelle est la différence avec le son originel ?

7. Grâce à la commande `subplot` et à la fonction *Func_ specgram.sci*, comparer le contenu temps-fréquence des deux signaux. Que remarquez-vous ?

8. Sachant que le taux de compression est égal au nombre de bandes conservées K divisées par le nombre total de bandes M, donnez le taux de compression obtenu avec cette méthode.