Université Paris-Est Marne-La-Vallée
Ingénieurs 2000. Option: Objects Communiquants

Matière: Traitement Numérique des Signaux
Novembre 2010

Projet de Traitement Numérique des Signaux

Proposé par: Prof. Abdellatif Zaidi

Proposé: 10 Nov. 2010 À rendre: Partie A : 23 Nov. 2010 Partie B : 03 Dec. 2010

Cet énoncé contient les deux parties (Partie A et Partie B) d'un projet à faire ce sémestre, dans le cadre du cours TSN. La première partie est à faire pour le 23 Novembre 2010, et la deuxième pour le 03 Décembre 2010. Le projet est à faire par tous les étudiants, et sera comptabilisé dans l'évaluation finale de ce cours.

Vous êtes demandés de me rendre vos rapports pour les deux parties. Chaque rapport devra contenir un résumé de votre approche, comment vous avez pu procéder pour résoudre le problème (de 1 à 5 pages). En plus des cinq pages au maximum, vous pouvez inclure toutes les figures Matlab que vous jugeriez utiles. Le nombre total de pages du rapport, y compris les figures, ne devra pas deṕasser 5 pages pour Partie A et 10 pages pour Partie B. Le nombre total de pages du rapport ne devra pas deṕasser 10 pages.

Chaque rapport recevra une note qui reflètera notre évaluation de votre compréhension des différentes problématiques abordées et l'effort que vous y mettez.

Note: Certaines commandes Matlab spécifiques seront nécessaires dans ce projet. Parcontre, par manque de temps, nous n'avons pas pu organiser une session Matlab qui aurait pu vous aider à mieux apprhender ce projet. Nous considèrerons donc que l'étudiant est suffisamment familier avec la plus part des commandes Matlab. Nous vous encourageons à commencer ce projets dès que vous pouvez.

Écouter du son avec MATLAB

À différents endroits de ce projet, il vous sera demandé de jouer des fichiers audio que vous auriez manipulés en utilisant Matlab. Alors que la fonction **sound** de Matlab fonctionne bien pour cela sur les plateformes MAC (OS X) et Linux (UNIX), l'utilisation de cette commande sur des machines Windows peut être un peu problématique dans certains cas et devra donc être évitée si cela est possible. Nous avons écrit différentes fonctions Matlab que vous pourriez utiliser au lieu de **sound** pour jouer des fichiers audio sur n'importe quelle machine. Ces fonctions **usound441** et **usound110** sont disponible sur la page de ce cours sur notre page Web. Elles permettent de jouer des séquences audio échantillonnées à 44100 Hz et 11025 Hz, respectivement.
Pour les utiliser, mettez les juste dans votre répertoire de travail ou tout autre répertoire qui est visible par votre installation Matlab. Pour des informations sur comment utiliser ces fonctions, tapez **help usound441** ou **help usound110**. Notez que vous pourriez avoir besoin de taper **add infoagents** dans votre shell ou prompt avant de lancer Matlab pour que ces fonctions fonctionnent proporement. Si vous rencontrez encore des difficultés à jouer des fichiers audio sous Matlab (après avoir essayé ce que nous vous avons suggéré), contactez le staff technique ou le responsable informatique.

Introduction

Un domaine d'application important des techniques de traitement numérique de signal est celui de traitement de la parole. Dans ce projet, nous considérons différents aspects liés à au filtrage et à la conception de filtres pour la parole. Plus spécifiquement, dans la Partie A, nous étudierons l'effet d'un filtrage passe-tout sur la parole. Vos connaissances actuelles dans le cadre de ce cours devront vous permettre de commencer cette partie immédiatement. Dans la deuxième partie, Partie B, nous essayerons de concevoir des filtres FIR et IIR dans le but d'améliorer la qualité d'un signal audio corrompu par un bruit additif. Ce projet est lié au contenu du chapitre sur la conception de filtres que nous sommes en train de finaliser.
Historiquement, les cours de traitement numérique des signaux ont donné une importance considérable aux techniques de conception de filtres FIR et IIR. Beaucoup d'alogorithmes de conception de filtres sont maintenant implémentés par des softwares tels que Matlab, ADS et Scialab. cela rend pas nécessaire que les praticiens apprennent les détails des algorithmes de conception; parcontre réléguer les fonctions de conception à un software rend la compréhension des propriétés des différents types de filtres optimaux, la signification des différents paramètres de conception et les compromis entre méthodes de conception encore plus indisponsable.

Dans le contexte de ce que nous étudions ce sémestre sur la conception de filtres FIR et IIR, nous vous suggérons de bien relire le cours et les examples étudiés, et de vous focaliser plus sur ce que sont que les filtres Butterworth et Chebyshev plutôt que les détails de la transformation bilinéaire.

Partie A (à rendre le 23 Nov. 2010)

Un résultat commun et assez "folklorique" stipule que l'oreille humaine est insensible à la phase, c'est à dire que les distortions de phases ne sont pas audibles. Si cela est correct, alors traiter un signal audio avec un filtre passe-tout ne devrait pas introduire de distortions qui peuvent être perçues. Cette partie de ce projet teste cette conjoncture. Le filtre passe-tout que nous consiérons et de la forme

$$H(z) = \left[\prod_{k=1}^{3} \frac{(z^{-1} - e_k^*)(z^{-1} - e_k)}{(1 - e_k z^{-1})(1 - e_k^* z^{-1})}\right]^N = \left[\frac{\sum_{k=0}^{6} b_k z^{-k}}{\sum_{k=0}^{6} a_k z^{-k}}\right]^N \tag{1}$$

Pour cette partie du projet, vous aurez besoin du fichier **projIA.mat**. Aprés avoir téechargé ce fichier à partir de la page Web du cours, copiez-le dans un répertoire visible depuis votre installation Matlab, et tapez **load projIA**. Les coefficients a_k et b_k seront stockées dans les vecteurs a et b, respectivement, $b(k+1) = b_k$ and $a(k+1) = a_k$.

(a) Implémentez le filtre passe-tout pour $N = 1$, et visualizez la réponse impulsionnelle (les 100 premiers échantillons), le module et la phase de la réponse en fréquence.

(b) Tracez le diagramme zéros-pôles. Vous pouvez vous aider de la fonction **roots** de Matlab.

(c) Filtrez le signal audio stocké sous la variable **speech** (échantillonné à $f_s = 11025$ Hz), en utilisant la commande **filter** pour utiliser le filtre implḿenté en (a). Écoutez le résultat et commentez plus spécifiquement sur oui ou non vous percevez une quelconque distortion audible.

(d) Implémentez le filtre pour $N = 50$ et, encore une fois, visualizer la réponse impulsionnelle (les 5000 premiers échantillons), le module et la phase de la réponse en fréquence.

(e) Répétez (b) et (c). Dans votre rapport, commentez plus spécifiquement comment vous pourriez décrire subjectivement la distortion et quel aspect du filtre en est la cause principale.

Partie B (à rendre le 03 Dec. 2010)

Pour cette partie du projet, vous aurez besoin du fichier **projIB.mat**. Aprés avoir téechargé ce fichier à partir de la page Web du cours, copiez-le dans un répertoire visible depuis votre installation Matlab, et tapez **load projIB**.
Dans cette partie du projet, des fonctions d'estimation d'ordre en Matlab (comme buttord, cheb1ord, ...) pourraient vous être utiles. Un petit probème avec l'utilisation de ces fonctions est que la définition de Matlab de "ripple" (erreur, ondulation) diffère pour les filtres FIR et IIR. Le paragraphe qui suit explique bien ces différences.

1. Pour la conception de filtres IIR, Matlab prend les spécifications en termes de 'ripple' sur l'amplitude et atténuation de la coupe-bande en tant que paramètres d'entrées. Pour une conception d'un filtre IIR, une erreur ou ripple sur l'amplitude δ_{IIR} résulte sur une conception avec un gain maximal unitaire en passe-bande (c'est à dire 0 dB) et un gain minimal de $10^{-\delta_{\text{IIR}}/20}$ (ou $-\delta_{\text{IIR}}$ dB). Une spécification d'une atténuation en coupe-bande de ξ_{IIR} résulte en une conception avec un gain maximal en coupe-bande de δ_{IIR} (ou $-\xi_{\text{IIR}}$ dB).

2. Pour la conception de filtres FIR aussi, Matlab prend une spécification du 'ripple' sur l'amplitude comme paramètre d'entrée. Une spécification de ce ripple de δ_{FIR} résulte en une conception avec un gain maximal en passe-bande de $(1+\delta_{\text{FIR}})$ (ou $20\log_{10}(1+\delta_{\text{FIR}})$ dB), et un gain minimal en passe-bande de $(1-\delta_{\text{FIR}})$ (ou $20\log_{10}(1-\delta_{\text{FIR}})$ dB). Une spécification d'une atténuation en coupe-bande de ξ_{IIR} résulte en une conception avec un gain maximal en coupe-bande de δ_{FIR} (ou $20\log_{10}(\xi_{\text{FIR}})$ dB).

Clairement, pour comparer des filtres FIR et IIR conçus en utilisant Matlab, nous devons trouver une correspondance entre nos spécifications désirées et les paramètres d'entrées des fonctions de conception de filtres. Soit donc G_{pbmax} (dB), G_{pbmin} (dB), et G_{sbmax} (dB) respectivement les gain maximal et minimal en passe-bande et le gain maximal en coupe-bande. Pour avoir plus de contrôle sur ces paramètres, nous introduisons aussi κ_{FIR} et κ_{IIR}, qui sont des termes de mise à l'échelle utilisés pour normliser nos filtres FIR et IIR.
Avant de commencer la partie B, répondez d'abord aux questions (a) et (b) suivantes.

(a) Trouvez δ_{FIR}, ξ_{FIR}, et κ_{FIR} en fonction de G_{pbmax}, G_{pbmin} , et G_{sbmax}. Ces paramètres seront utilisés dans la conception de filtres FIR.

(b) Trouvez δ_{IIR}, ξ_{IIR}, et κ_{IIR}, en fonction de G_{pbmax}, G_{pbmin}, et G_{sbmax}. Ces paramètres seront utilisés dans la conception de filtres IIR.

La partie B de ce projet concerne la conception d'un filtre passe-bas pour l'élimination des composantes hautes-fréquences d'un bruit entiâchant un signal audio. Le signal bruité est stocké sous la variable **noisy** et a été échantillonné à 44100 Hz. Ce signal bruité consiste en la somme d'un signal audio à bande limitée à 4kHz obtenu avec un filtre passe-bas à bande de transition très étroite, et d'un bruit filtré à 4kHz à l'aide un filtre passe-haut à bande de transition très étroite. Afin de d'éliminer le bruit du signal bruité, nous devons concevoir un filtre numérique avec les spécifications suivantes:

1. Bande passante : $f < 2500$ Hz.
2. Coupe bande : $f > 4000$ Hz.
3. Gain maximal en bande passante G_{pbmax}: 40 dB.
4. Gain minimal en bande passante G_{pbmin}: 37 dB.
5. Gain maximal en coupe bande G_{sbmax}: -55 dB.

Vous auriez probablement besoin de quelques itérations avant d'aboutir à la conception finale voulue de chaque filtre. Pour cette raison, nous vous suggérons d'écrire un script matlab séparé pour chacun des filtres à concevoir.

Notez aussi que, dans cette partie, les spécifications fixées sont assez sévères et donc vos filtres pourraient avoir des ordres assez élevés. Vous pourriez essayer d'implémenter les filtres IIR à l'aide de commandes Matlab existantes (comme **fdatool**, **butter**, **cheby1**, **cheby2**, ou **ellip**–tapez "**help signal**" pour plus d'informations). Si vous faites ainsi, vous allez remarquer que les filtres que vous obtenez peuvent ne pas être stables. Il y a différentes raisons à cela, la principale étant liée à la quantification des coefficients. Cela intervient ici parceque les spécifications sont assez strictes et certains des filtres auront leurs pôles concentrés près de $z = 1$.
Pour remédier à ce problème, vous pourriez regrouper les pôles et les zéros pour le filtre voulu en paires (conjugués ou pas) pour créer des filtres de second ordre qui sont stables, de la forme $N(z)/D(z)$ où $N(z)$ et $D(z)$ sont des polynômes en z d'ordres inférieurs ou égaux à deux. Une mise en cascade de tels sous-systèmes produira donc le système désiré. L'incovenient est que vous devriez implémenter vos propres méthodes pour générer les figures et les simulations.

B1) Pour chacun des types de filtres suivants, concevez un filtre numérique qui satisfait les spécifications fixées plus-haut:

Butterworth, Chebyshev Type I, Chebyshev Type II, Elleptic, Parks-McClellan, Kaiser

Pour chaque filtre conçu, précisez ce qui suit dans votre rapport.

(a) L'ordre du filtre.

(b) Le nombre de multiplications par échantillon d'entrée nécessaires pour implémenter le filtre. Exliquez la structure que vous utilisez.

(c) Tracez l'amplitude de la réponse fréquencielle $|H(e^{jw})|$ (en dB) pour $0 \leq w \leq \pi$ en utilisant **freqz**.

(d) Tracez le diagramme zéros-pôles.

(e) Tracez la réponse impulsionnelle en utilisant **filter** et **stem** pour 100 échantillons. (Utilisez **subplot** pour mettre le diagramme zéros-pôles et la réponse impulsionnelle sur la même figure.)

(B2) Filterez le signal **noisy** en utilisant le filtre conçu. Écoutez le signal filtré et le signal original bruité et comparez les deux.