L3 MIA / Systèmes de Communications

TP : Tranmission de la voix sur une liaison GSM

Lancement

1. lancez une fenêtre terminal
2. tapez /users/ufr/prof/maheg/lanceGSM.sh

Préliminaires

Ce TP utilise scilab, un logiciel libre de simulation numérique (voir `http ://www.scilab.org`). On peut soit taper directement les instructions dans la fenêtre de commande qui apparaît, soit exécuter un script stocké dans un fichier *.sce*, en ouvrant le fichier dans l'éditeur intégré (première icône à gauche) et en utilisant la commande *Exécuter/charger dans scilab*). On peut afficher l'aide en ligne sur une fonction par `help` suivi du nom de la fonction.

Scilab est un langage vectoriel : les données sont représentées par des vecteurs ou des matrices (tableaux à 1 ou 2 dimensions, respectivement) qui sont traités en bloc (éviter les boucles échantillon par échantillon). Vous commencerez par tester ligne par ligne (surligner puis cliquer avec le bouton du milieu dans la fenêtre de commande) les exemples du fichier *exemples.sce*, pour vous familiariser avec la manipulation des vecteurs en scilab.

Pour écouter un fichier son au format GSM, vous utiliserez la fonction *play_gsm* fournie dans le répertoire : `play_gsm("toto.gsm")` . N'oubliez pas auparavant de régler le contrôleur de volume.

1 Impact des erreurs selon la classe des bits

Ouvrez le fichier *TP.sce*. Ce programme simule la transmission d'une séquence de parole issue d'un codeur GSM à travers un canal bruité :

- lecture du fichier d'entrée, au format GSM, par trames de 264 bits (260 + 4 bits de bourrage = 33 octets) ;
- ajout d'une erreur binaire par le canal ;
- écriture des trames de réception dans le fichier de sortie.

Chaque fonction appelée est définie dans un fichier portant son nom suivi de l'extension *.sci*.

Ouvrez le fichier *canal_binaire_1bloc.sci*, qui simule un canal binaire symétrique de paramètre p (probabilité d'erreur sans codage) concentrant les erreurs sur une seule des 3 classes de bits. Dans la version fournie, la classe II est touchée, *via* les réglages suivants :

- `p_bis = p*260/78;`
- `err_bin(1:182)=0;`

Cette fonction calcule d'abord un vecteur d'erreurs uniformément réparties de probabilité p^{bis}, annule cette erreur aux positions non concernées (1 à 182 ici) et ajoute l'erreur à la trame d'entrée.

Exécutez le programme *TP.sce* et écoutez le fichier de sortie. Faites les modifications nécessaires pour simuler des erreurs concentrées sur la classe Ib, exécutez le programme en veillant à changer le nom du fichier son de sortie et comparez la qualité audio avec celle obtenue précédemment. Idem avec les bits de la classe Ia. Il se peut que la fonction *play_gsm* ne fonctionne pas pour la classe Ia ; dans ce cas ouvrez et écoutez le fichier avec *audacity*.

2 Codage de canal

Nous allons maintenant insérer avant le canal l'opération de codage. Constituez tout d'abord, à partir de la trame d'entrée, un vecteur par classe de bits à partir du vecteur trame. Comme une erreur non détectée sur la classe Ia d'une trame rend tout le fichier son inutilisable, le codage en bloc de la classe Ia sera remplacé par 3 bits supplémentaires à 0 et la détection d'erreur sera simulée par comparaison directe entre la classe Ia reçue et celle émise (ce qui n'est naturellement pas réaliste). Effectuez le codage convolutif de la classe I (sans oublier les 4 bits de queue) en utilisant la fonction *codeur_convolutif* fournie. Vous appellerez *trame_codee* la trame de 456 bits ainsi constituée.

Modifiez *canal_binaire_1bloc.sci* pour que les erreurs soient réparties sur tous les bits. Vous appelerez *trame_codee_out* la trame de sortie du canal.

A la sortie du canal, désassemblez la trame et décodez la classe I en utilisant la fonction *decod_Viterbi* fournie. Vous appelerez *trame_out_tmp* la trame ainsi décodée. N'oubliez pas de lui ajouter les 4 derniers bits de *trame* (bits de bourrage).

On simule la détection d'erreur sur la classe Ia en comparant la classe Ia reçue à celle émise. Vous utiliserez pour cela la fonction *d_Hamming* fournie. Si une erreur est détectée, alors on utilise la classe Ia de la trame précédente (sauf pour la 1ère : utiliser la trame émise). Vous appelerez *trame_out* la trame ainsi corrigée. N'oubliez pas de lui ajouter les 4 derniers bits de *trame* (bits de bourrage).

Complétez le programme de manière à calculer le taux d'erreur. Celui-ci sera s'appuiera sur la comparaison entre *trame* et ***trame_out_tmp***.

Simulez en changeant le nom du fichier de sortie, comparez la qualité audio à celles obtenues précédemment, comparez le taux d'erreur à p (10^{-2}). Recommencez avec $p = 10^{-1}$. Conclusion ?

3 Entrelacement

Jusqu'à présent, nous avons simulé un canal avec des erreurs dispersées. Les erreurs sur le canal radio-mobile sont en fait groupées par salves de 10 à 100 bits. Vous simulerez ce canal en utilisant la fonction *[bloc_out,nb_err_apres] = canal_binaire_1bloc_salves(bloc_in,Pe,nb_err_avt)* fournie. Comme une salve d'erreurs sur une trame peut se prolonger sur la trame suivante, cette fonction a deux paramètres de plus que la précédente : *nb_err_avt* indique le nombre d'erreurs issues de la trame précédente se pousuivant sur la trame courante ; *nb_err_apres* indique le nombre d'erreurs issues de la trame courante se poursuivant sur la trame suivante.

1) Simulez avec $p = 10^{-2}$. Comparez le taux d'erreur au taux précédent (pour $p = 10^{-2}$) et à p.

2) Faites un entrelacement / désentrelacement intra-trame seulement (matrice 8x57). Simulez et notez la variation du taux d'erreur.

3) Mettez en œuvre l'entrelacement complet du GSM.