SUJET DE TP I4-FC2

PROPOSÉ PAR P.JARDIN (G.BAUDOIN)

DECODAGE DE CODE CONVOLUTIF PAR L'ALGORITHME DE VITERBI

OBJECTIFS DU TP (DURÉE : 4HEURES)

L'objectif de ce TP est d'implanter sous matlab un décodage de code convolutif en utilisant l'algorithme de Viterbi. On utilisera la distance de Hamming, c'est à dire qu'on effectuera un *hard decoding.*

Logiciel utilisé :

On utilise pour ce TP le logiciel matlab. Pour obtenir de l'aide sur une fonction matlab, il suffit de taper la commande ***help*** suivie du nom de la fonction.

TRAVAIL A EFFECTUER

Ecrire un script principal qui permette de tester votre algorithme de décodage (deuxième étape) avec le code convolutif utilisé sur les liaisons GSM.

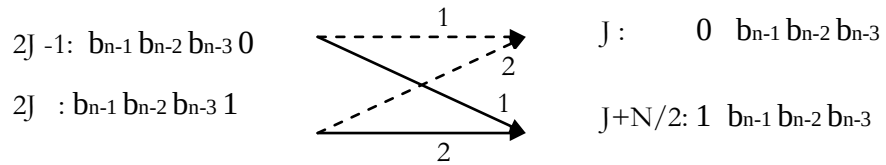
Il s'agit d'un code de taux 1/2 (k=1, n=2) dont les 2 polynômes générateurs sont $g_1(x)= 1+x^3+x^4$ et $g_2(x)= 1+x+x^3+x^4$. En octal g_1 s'écrit 23 et g_2 33.

Vous devez donc écrire un script principal qui :

- génère la matrice des sorties de 2 bits correspondant aux N=16 états possibles pour une entrée nulle (***mat_sort***)
- génère la matrice etat_prec des états pouvant précéder un état donné (en fonction de l'état actuel et de la transition 1 ou 2 choisie pour aboutir à cet état actuel (voir indications et schéma du treillis ci-dessous).
- génère une suite de bits (***randint***) : (dans une première étape vous pourrez prendre celle considérée en TD, ensuite vous pourrez tester avec des suites de longueur égale à celle des trames GSM c'est-à-dire 185 (+4 zéros) (voir indications sur le codeur GSM plus loin)
- génère la suite codée correspondante (***codconv***)

Indications :

On pourra utiliser le fait que dans le treillis d'un code de taux 1/n, les états de départ 2J -1 et 2J aboutissent aux états J et J+N/2, où N est le nombre d'états et **J varie de 1 à N/2** (dans le cas du codeur utilisé dans le GSM N=16) . Attention par rapport au TD il y a une différence d'indice pour numéroter les états : on appelle état 1 l'état où tout le registre de mémoire est à 0.



Écrire sous Matlab l'algorithme de Viterbi. Il comporte deux parties :

1) Le calcul de la distance cumulée minimale D_{cum} pour arriver à chaque instant t (de 1 à T) en tout état i (de 1 à N) avec sauvegarde de la décision (dec) prise.

$$[D_{cum}(t,i), \mathrm{dec}(t,i)] = \min_{j \text{ précédant } i} \left(D_{cum}(t-1, j) + d_H\left(r_t, c(j,i)\right)\right)$$

Vous devrez donc écrire aussi une fonction simple qui calcule la distance de hamming d_H entre les deux bits reçus au temps t et les deux bits codés émis sur la transition j vers i (ceux-ci s'obtiennent à partir de la matrice retournée par ***mat_sort*** et des indications qui suivent).

Indications :

Si $c_1\ c_2$ sont les 2bits codés obtenus sur une branche en traits pointillés partant d'un état, les deux bits codés sur la branche en traits pleins correspondante sont $\overline{c}_1\ \overline{c}_2$ (en matlab ~1=0 et ~0=1)

2) La remontée du chemin optimal pour déterminer la séquence de bits d'entrée la plus vraisemblable.

Initialisation : $état(T) = 0$ si tail bits ou $= \arg\min(D(T,:))$ sinon .

Puis à tout instant (en remontant le temps)

$$état(t-1) = \text{etat\_prec}\left(état(t), dec\left(t, état(t)\right)\right)$$

$$bit(t) = \text{bit\_input}\left(état(t)\right)$$

- Le calcul du bit d'entrée correspondant à l'état actuel est très simple à effectuer à partir des indications précédentes (schéma du treillis).

Tester ce programme

Vous devez donc lancer l'algorithme de Viterbi dans le script principal pour obtenir la séquence décodée à partir de la séquence reçue.

Dans un premier temps la séquence reçue sera prise égale à la séquence codée (cas sans erreur). Puis vous pourrez sur une séquence courte tester des configurations à 1, 2 ou 3 erreurs. (on rappelle que la distance libre est égale à 7, voir TD).

- Vous pourrez modifier à la main un deux ou trois bits de la séquence codée avant de lancer l'algorithme de Viterbi sur cette séquence reçue avec erreurs.

Enfin pour tester les « vraies » performances du décodeur vous intégrerez les étapes suivantes dans votre script principal en ayant codé des trames GSM de 185 bits + 4 zéros:

1) avant de lancer l'algorithme de Viterbi
   - génére une séquence d'erreur (de même taille que la suite codée c'est-à-dire 378 bits) avec une certaine probabilité d'erreur p (***randsrc***). On prendra successivement $p=10^{-3}$, $p=10^{-2}$, $p=10^{-1}$
   - additionne modulo 2 cette séquence d'erreur à la suite codée pour former la séquence reçue.
2) Après l'algorithme de Viterbi
   - calcule le nombre d'erreurs entre la séquence décodée et la suite de bits initiale.
3) Pour obtenir un vrai taux d'erreur il faudra lancer un grand nombre de fois le script obtenu

Indications sur le codeur GSM :

Le codeur de parole fournit 260 bits toutes les 20ms, ce qui correspond à 13 Kbps. Ce train binaire est encodé en fonction de l'importance des bits en sortie du codeur de parole. Les 50 bits les plus importants sont codés par un code cyclique (53, 50, 2). Puis ils sont concaténés avec les 132 d'importance juste inférieure. Et le code est appliqué sur ces groupes de 53 +132 = 185 bits auxquels on rajoute 4 tail bits de façon à être sur d'arriver dans l'état « 0 0 0 0 » à la fin d'un slot. On encode donc 189 bits ce qui donne 378 bits après codage convolutif. À ces 378 bits sont ajoutés les 78 bits restant qui ne sont pas protégés. On obtient ainsi un bloc de 456 bits toutes les 20 ms. Le débit binaire après codage est donc de 22,8 Kbps.