ESIEE : Signaux et Télécommunications

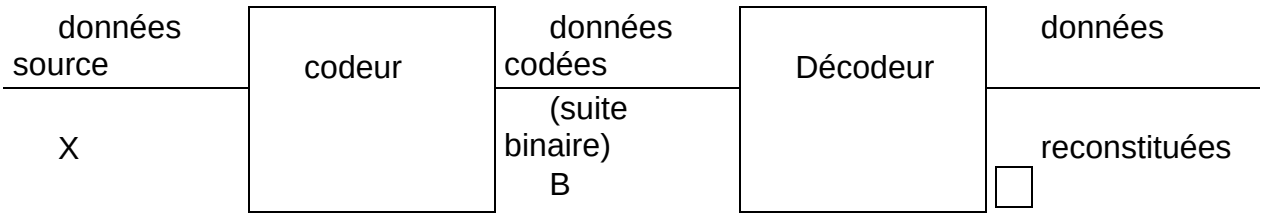
Codage de source

Introduction

Le codage de source ou compression des données sert à fournir une représentation efficace des données (un taux de compression important) tout en préservant l'information essentielle qu'elles portent. Il est employé pour le stockage ou la transmission de ces données (on appelle données le résultat de la numérisation de signaux comme ceux de parole ou d'images ou plus généralement les données disponibles sur un fichier d'ordinateur) .

Le codage de source est d'autre part connecté à d'autres applications techniques telles que la classification d'images, la reconnaissance vocale,...

Le bloc diagramme d'un système de compression est le suivant :



La théorie de l'information développée par Shannon dans les années 1940-1950 fournit des idées fondamentales pour la conception d'un grand nombre de techniques de codage de source. Une idée est que des messages numériques peuvent être comprimés en associant les mots de code les plus courts aux messages les plus probables et que le taux de compression (sans perte) maximum que l'on peut atteindre peut être déterminé à partir d'une description statistique des messages. Cette stratégie conduira à un codage à débit variable et à des techniques de codage sans perte (ou codage entropique) c'est-à-dire telles que les données reconstituées soient identiques aux données source.

Une autre idée est que les systèmes de codage sont plus performants lorsqu'ils opèrent sur des vecteurs (ensemble d'échantillons) plutôt que des échantillons individuels. Cette idée a été indirectement exploitée dans les années 70 lorsque l'on a effectué des pré-traitements tels que la prédiction ou une transformation avant de quantifier scalairement les échantillons. Elle a ensuite été pleinement utilisée dans les années 80 avec les techniques de quantification vectorielle. Toutes ces techniques (qui opèrent une quantification des données) font partie des techniques de compression avec perte. Les données reconstruites ne coïncident pas avec les données source mais sont (idéalement) perceptuellement non distinctes de celles-ci. La plupart du temps le système obtenu pour ces techniques est à débit fixe.

Exemple :

On cherche à comprimer l'image 4x4 suivante où chaque pixel est caractérisé par une couleur parmi 4 : R = « rouge », J = « jaune », B = « bleu », V = « vert » .

R	R	R	R
R	B	R	B
R	B	V	B
R	J	J	J

Codage sans perte

La méthode triviale pour transmettre cette information sous forme binaire est d'associer un mot de code sur 2 bits à chaque couleur.

Ex : R = « 00 » B = « 01 » V = « 10 » J = « 11 »

l'image peut être codée sur 32 bits en parcourant les lignes de l'image de haut en bas et de gauche à droite :

00 00 00 00, 00 01 00 01, 00 01 10 01, 00 11 11 11

Une méthode plus efficace tiendra compte des probabilités de chaque couleur en opérant l'affectation de mots de taille différente à chaque couleur

R = « 1 » B = « 01 » J = « 001 » V = « 000 »

ce qui conduit à coder l'image sur 28 bits.

1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 0 1 0 0 1

Codage avec perte

Pour chaque sous bloc 2x2 de l'image on garde la couleur dominante, c'est-à-dire :

R	R
R	J

ce qui donne après le codage « trivial »

00 00 00 11 → 8 bits

et après le codage « entropique »

1 1 1 001 → 6 bits

mais bien sur on ne peut pas revenir à l'image initiale avec ce codage.

Critères d'appréciation d'un codage :

Pour un codage sans perte ou cherchera à maximiser le taux de compression et à minimiser la complexité de mise en œuvre et le retard introduit.

Pour les codages avec perte on cherchera aussi à maximiser le taux de compression et à réduire la complexité mais aussi à avoir le moins de perte possible.

Cette « perte » sera appréciée de façon objective par des mesures de distorsion ou de rapport signal sur bruit. Pour des signaux audio ou vidéo on évaluera la qualité subjective des données reconstituées et on calculera des grandeurs objectives tenant compte de nos capacités de perception (RSB perceptuel).

Codage et quantification :

Le codage consiste souvent (cas des codages avec perte) en deux opérations successives.

La première opération est une opération de quantification qui consiste à discrétiser l'amplitude des données (prises isolément ou en bloc, après transformation prédiction ou non) sur un ensemble fini de valeurs $\{y_i\}$ i=1...L.

La deuxième opération qui est le codage proprement dit (c'est-à-dire la traduction sous forme binaire d'un message de données) consiste à traduire en suite binaire l'indice i du représentant y_i choisi. Lorsque $L = 2^b$ on peut transmettre le résultat de la quantification sur b bits.

Les techniques de codage sans pertes s'utilisent sur des données pouvant prendre un nombre fini de valeurs possibles , en particulier sur des données préalablement quantifiées. Ainsi le codage entropique, présenté dans la première partie de ce cours sera généralement appliqué en pratique en aval d'un codage avec perte (et quantification).

1. Codage entropique – codage sans pertes

1.1 Introduction

Dans certains systèmes de communication, un débit fixe n'est pas toujours le plus approprié car la source d'information présente une activité très variable (silences dans un signal de parole, blancs dans un fac-simile). Un codage à débit variable qui peut ajuster son débit pour s'adapter à l'activité locale (ou momentanée) de la source est alors idéal mais généralement non directement utilisable sur un support de transmission classique. Pour utiliser un tel support à débit fixe ,le débit moyen à long terme doit être constant et des buffers sont nécessaires pour adapter dans les deux sens le débit variable à court terme et le débit fixe à long terme.

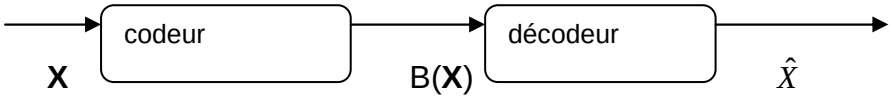
Ces buffers ajoutent de la complexité au système et doivent résoudre les problèmes d'underflow et overflow (pas assez ou trop de données pour le lien à débit fixe).

Les systèmes de stockage sont intrinsèquement à taux variable de même que certains systèmes de communication (par paquets).

1.2 Système de codage sans perte

Dans un système de compression sans perte, le décodeur est capable de reconstruire exactement les données source.

Le bloc diagramme est le suivant :



X représente le vecteur aléatoire de données source à comprimer ,c'est une séquence de taille finie **X**=(X_1,X_2 ,… ,X_n)(n peut être arbitrairement grand).

Les échantillons X_i de **X** appartiennent à un alphabet fini A={a_1,a_2,…,a_M)
(il y a déjà eu une quantification)

Exemples :

- Suite d'échantillons de chrominance et luminance des pixels d'une image
- texte (lettres de l'alphabet (26) et signes de ponctuation)

La sortie du codeur est représentée par la séquence binaire B(**X**)=(b_1,b_2,…,b_k) dont la longueur k est variable (en fonction des éléments de **X** et pas seulement en fonction de n)

B(**X**) est le mot de code assigné à **X**. Étant donné que le système est sans perte le codeur doit assigner des mots de code distincts à des vecteurs de données distincts.

Le taux de compression est $r=n\log_2(M)/k$ où M est la taille de l'alphabet A.
La résolution est R=k/n (bits par échantillons)
Ces deux valeurs peuvent être variables mais de moyenne (statistique) constante.

1.3 Codage d'une source discrète sans mémoire

Pour une source sans mémoire les échantillons X_i du vecteur de données **X** à coder correspondent à des variables aléatoires indépendantes et identiquement distribuées (suite i.i.d.) .
Dans ce cas chaque échantillon X_i est codé (remplacé par un mot de code qui est une suite binaire).
Le codeur fait correspondre à chaque symbole d'entrée x (réalisation d'une v.a. X dont les valeurs possibles appartiennent à l'alphabet A) un mot de code binaire c(x) de longueur l(x) (qui est en fait la longueur de c(x)).
Le décodeur opère la correspondance inverse et associe à des suites binaires b des symboles y appartenant à A : d(b)=y de sorte que **d(c(x))=x (codage sans perte)**

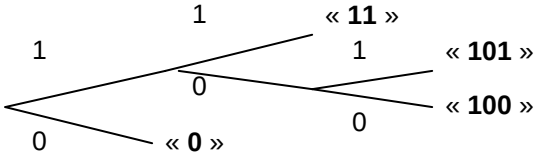
Le but du codage est de garder un nombre de bits transmis aussi faible que possible c'est à dire que l'on cherche à minimiser la longueur moyenne du code :

$$\bar{l} = E(l(X)) = \sum_{a \in A} p(a)l(a)$$

1.3.1 Condition du préfixe :

D'autre part le code n'est utile que si on sait le décoder sans ambiguïté c'est à dire si pour une suite binaire codée de longueur finie il n'y a qu'une séquence de symboles d'entrée possible. Pour satisfaire cette condition sans introduire de séparateurs entre les mots de code , **le code doit satisfaire la condition dite du préfixe qui stipule qu'aucun mot de code ne doit être le préfixe d'un autre mot de code.**

Exemples : {0,10,101,0101} n'est pas un code à préfixe
{0,10,110,111} est un code à préfixe
La condition du préfixe assure l'unicité du décodage du code
Les codes binaires à préfixe peuvent être représentés par un arbre binaire tel qu'on le voit sur la figure suivante :



Les mots de code sont les suites de labels(‘0’ ou ‘1’) entre le nœud racine et les nœuds terminaux (feuilles)de l’arbre

Par construction de l’arbre , aucun mot de code ne peut être le préfixe d’un autre mot .

Un codage est l’opération d’assignation des différents mots de code aux différents symboles de source (éléments de A)

1.3.2 Inégalité de Kraft

La plus importante propriété des codes uniquement décodables (codes à préfixe) est donnée par l’inégalité de Kraft :

Théorème 1: Une condition nécessaire et suffisante pour qu’un code correspondant à la source d’alphabet A={a_1,a_2,…,a_M) et constitué de mots de code de longueur l_k =l(a_k) (k=1…M) soit uniquement décodable est que

$$\sum_{k=1}^{M} 2^{-l_k} \le 1$$

Démonstration $\Rightarrow$:

Étant donnée une séquence d’entrée de K symboles X=(x_1,x_2,…,x_K) de longueur totale L = l(X) = l(x_1)+l(x_2)+…+l(x_K),

Si on note N(L) le nombre total de séquences X de taille K qui sont codées par une séquence binaire de longueur L

Si l_{max} représente la longueur maximale des mots de code : l_{max} = max (l_k ; k=1…M)

On a L<= Kl_{max} = L_{max}

Si le code est décodable de façon unique alors les N(L) séquences d’entrée doivent générer des séquences binaires distinctes de longueur L

Puisqu’il y a 2^L séquences binaires distinctes de longueur L nous devons avoir :

N(L)<=2^L

$$\left(\sum_{k=1}^{M} 2^{-l_k}\right)^K = \left(\sum_{a\in A} 2^{-l(a)}\right)^K$$

$$= \sum_{a_1\in A}\sum_{a\in A_2}\cdots\sum_{a_K\in A} 2^{-\sum_{i=1}^{K} l(a_i)}$$

$$= \sum_{L=1}^{L_{\max}} N(L)\, 2^{-L}$$

$$\le L_{\max}$$

donc

$$\sum_{k=1}^{M} 2^{-l_k} \le L_{\max}^{1/K} = K^{1/K} l_{\max}^{1/K}$$

et puisque la borne est valable quelque soit K, on peut prendre la limite quand K tend vers l’∞

ce qui donne :

$$\sum_{k=1}^{M} 2^{-l_k} \leq 1$$

Démonstration ⇐ :

On peut supposer sans perte de généralité que les longueurs des mots de code sont ordonnées :

$l_1 \leq l_2 \leq \ldots \leq l_M$

Un mot de code de longueur l correspond à un trajet de l branches dans l'arbre binaire commençant au noeud racine et se terminant à un nœud de profondeur l

Étant donné un ensemble de longueur l_k satisfaisant l'inégalité de Kraft :

Considérons comme premier mot de code c_1 une séquence binaire correspondant à un trajet de l_1 branches

Comme aucun mot de code ne peut avoir ce mot de code comme préfixe , on coupe la branche après le nœud terminal correspondant

Prenons ensuite un trajet disponible de profondeur l_2 .(Il y a 2^{l2}-2^{l2-l1} possibilités) et coupons la branche après le nœud correspondant.

Il reste pour le troisième mot de code 2^{l3} - 2^{l3-l2} – 2^{l3-l1} possibilités

Pour le m ième mot de code on choisit un trajet parmi les n(m) possibilités

$$n(m) = 2^{l_m} - \sum_{k=1}^{m-1} 2^{l_m - l_k} = 2^{l_m}(1 - \sum_{k=1}^{m-1} 2^{-l_k})$$

si m≤M

$$n(m) = 2^{l_m}(1 - \sum_{k=1}^{M} 2^{-l_k} + \sum_{k=m}^{M} 2^{-l_k}) \geq 2^{l_m} \sum_{k=m}^{M} 2^{-l_k}$$ d'après l'inégalité de Kraft

et cette dernière quantité étant supérieure ou égale à 1 , il existe un mot de code de la longueur l_m souhaitée .

Ceci prouve que l'on peut trouver un code à préfixe étant donné un jeu de longueurs obéissant à l'inégalité de Kraft .

Cette inégalité fournit la base théorique pour calculer des bornes inférieure et supérieure à la longueur moyenne d'un code uniquement décodable

1.3.3 Bornes pour la longueur moyenne d'un code

Avant de préciser ces bornes, examinons une proposition utile pour leur calcul.

Proposition :

Étant données deux distributions de probabilité p et q pour un alphabet commun A (p(i) = proba (X = a_i)) alors on définit la distance de Kullback - Leibler ou l'entropie relative entre les distributions p et q par :

$$D(p \,//\, q) = \sum_{i=1}^{M} p(i) \log_2 \frac{p(i)}{q(i)}$$

On montre que cette quantité est positive ou nulle et s'interprète comme une mesure de distance entre les deux distributions.

Démonstration :

On utilise : $\forall x > 0 \qquad \log_2 x \le (x-1)\log_2 e$

en choisissant $x = \dfrac{q(i)}{p(i)}$

$$\log_2 \frac{q(i)}{p(i)} \le \left(\frac{q(i)}{p(i)} - 1 \right) \log_2 e$$

$$\Leftrightarrow \log_2 \frac{p(i)}{q(i)} \ge \left(1 - \frac{q(i)}{p(i)} \right) \log_2 e$$

$$\Rightarrow D(p \,//\, q) \ge \sum p(i) \log_2 e - \sum q(i) \log_2 e$$

$$\Leftrightarrow D(p \,//\, q) \ge 0$$

L'inégalité de Kraft et la proposition précédente sur la distance de Kullback permettent de démontrer les théorèmes suivants :

Théorème 2 : Étant donné un code uniquement décodable opérant sur une source X qui peut prendre ses valeurs dans l'alphabet fini A avec la distribution de probabilité p alors la longueur moyenne de ce code vérifie :

$$\overline{\ell} \ge H(X) \qquad \textbf{où H(X) est l'entropie de la source X}$$

Démonstration :

Le code uniquement décodable vérifie l'inégalité de Kraft.

$$\sum_{k=1}^{M} 2^{-\ell k} < 1$$

On crée une nouvelle distribution de probabilité de la forme

$$q(i) = a 2^{-\ell i} \left(\text{avec } a \ge 1 \text{ pour avoir } \sum_i q(i) = 1 \right)$$

La proposition dit que :

$$\sum p(i) \log_2 \frac{p(i)}{a 2^{-\ell i}} \ge 0$$

$$\Leftrightarrow H(X) \le -\sum p(i) \log_2 a + \overline{\ell}$$

$$\Leftrightarrow H(X) \le \overline{\ell}$$

Théorème 3 : Il existe un code uniquement décodable pour la source X de loi p tel que la longueur moyenne du code vérifie

$$\overline{\ell} < H(X) + 1$$

Démonstration :

Prenons la longueur ℓ_k comme l'entier satisfaisant

$$2^{-\ell_k} < p_k < 2^{-\ell_k+1}$$

alors $\quad -\ell_k \le \log_2 p_k < \ell_k + 1$

$\Leftrightarrow \quad -\log_2 p_k \le \ell_k < -\log_2 p_k + 1$

(l_k est le plus petit entier supérieur ou égal à $-\log_2 p_k$ et le code obtenu de la sorte s'appelle code de Shannon).

$\overline{\ell} = \sum_k \ell_k p_k$ vérifie alors de façon évidente

$H(X) \le \overline{\ell} < H(X) + 1$

par ailleurs

$$\sum_{k=1}^{M} 2^{-\ell_k} \le \sum_{k=1}^{M} 2^{\log_2 p_k} = 1$$

L'inégalité de Kraft est donc vérifiée, on peut donc effectivement construire ce code.

Remarque : Ce sont ces deux inégalités : $H(X) \le \overline{\ell} < H(X) + 1$

(bornes pour la longueur moyenne des mots d'un code) qui sont à l'origine du terme «codage entropique » pour le codage sans perte uniquement décodable.

Exemple :codage d'un texte en français :

Les 26 lettres de l'alphabet plus l'espace ¤ utilisées pour un texte écrit (sans accents ni points de ponctuation) n'ont pas la même probabilité d'apparition :

P('e')>..>P('a')>…>P('w')>…

On peut calculer ces probabilités par la mesure des statistiques d'utilisation de chaque lettre sur une grande base de données .

Lorsqu'on connaît l'entropie H(<$\log_2(27)$) on sait que le meilleur code possible pour ces lettres aura une longueur moyenne supérieure ou égale à H

1.3.4 Algorithme de Huffmann.

Cet algorithme consiste à construire progressivement l'arbre binaire correspondant à un code à préfixe en partant des nœuds terminaux.

On part des listes $\{a_1 ... a_M\}$ et leur probabilité $\{p_1 ... p_M\}$

On sélectionne les deux symboles les moins probables, on crée deux branches dans l'arbre et on les étiquette par les deux symboles 0 et 1.

On actualise les deux listes en rassemblant les deux symboles utilisés en un nouveau symbole et en lui associant comme probabilité la somme des deux probabilités sélectionnées.

On recommence les deux étapes précédentes tant qu'il reste plus d'un symbole dans la liste.

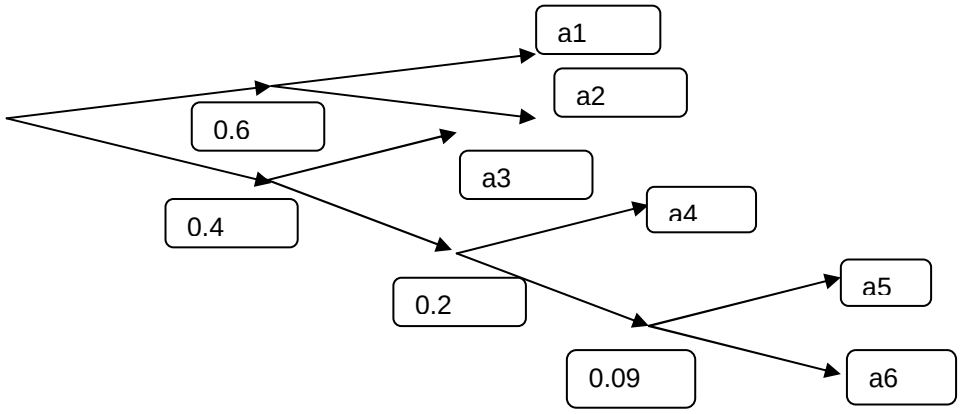
On montre que cet algorithme est optimal. Pour aucun autre code uniquement décodable la longueur moyenne des mots de code n'est inférieure à celle obtenue pour cet algorithme .

Exemple :

Symboles	a_1	a_2	a_3	a_4	a_5	a_6
Probabilités	0,35	0,25	0,2	0,11	0,05	0,04

H(X) = 2,25

$\overline{\ell} = 2,29$



Cet algorithme est utilisé dans la norme MPEG de codage audio et vidéo (pour coder les coefficients de la DCT (Discrete Cosinus Transform)).

1.4 Généralisation - codage entropique vectoriel

La double inégalité $H(X) \le \overline{\ell} < H(X)+1$ est trop imprécise car H(X) est généralement faible. On peut diminuer cette imprécision en appliquant les résultats précédents à une source « étendue » c'est-à-dire en formant des vecteurs aléatoires X_N groupant N variables aléatoires $X_N = (X^{(0)}, X^{(1)}, ..., X^{(N-1)})$

L'entropie de la source est alors

$$H(X_N) = -\sum_{a^N} p_{X_N}(a^N) \log_2(p_{X_N}(a^N))$$

a^N est une notation pour représenter un N-uplet de valeurs prises dans l'alphabet A.

Il y a M^N possibilités pour la valeur de a^N.

Les probabilités sont généralement difficiles à établir sauf dans le cas où les $X^{(i)}$ sont indépendantes (source i.i.d.), auquel cas :

$$p_{X_N}(a^N) = \prod_{a \text{ de } a^N} p_X(a) \quad \text{et} \quad H(X_N) = NH(X)$$

Si on associe un mot de code à chaque groupe de N symboles a^N, alors on a montré qu'il existe un code dont la longueur moyenne vérifie

$$H(X_N) \le \bar{\ell}_N < H(X_N) + 1$$

$$\left(\text{avec } \bar{\ell}_N = \sum_{a^N} p_{X_N}(a^N) \ell(a^N) \right)$$

Le nombre de bits moyen par symbole $\bar{\ell} = \frac{\bar{\ell}_N}{N}$ vérifie alors :

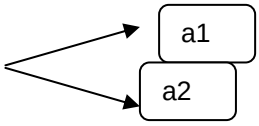
$$\frac{H(X_N)}{N} \le \bar{\ell} < \frac{H(X_N)+1}{N}$$

soit pour une source i.i.d.

$$H(X) \le \bar{\ell} < H(X) + \frac{1}{N}$$

Exemple : Supposons une source ne pouvant prendre que deux valeurs a_1 et a_2 de probabilité $p_1 = 0,3$ et $p_2 = 0,7$

si on code cette source on aura

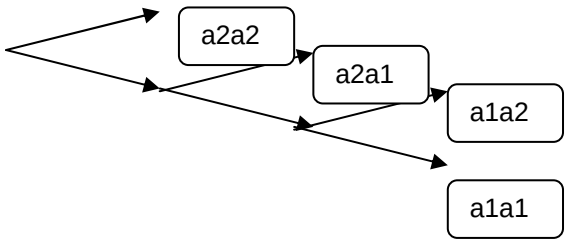


et $\bar{\ell} = 1$

si on forme des groupes de 2 valeurs

a_1a_1 a_1a_2 a_2a_1 a_2a_2

0,09 0,21 0,21 0,49



$$\ell_2 = 0,49 + 0,21.2 + 0,3.3$$

$$\bar{\ell} = 0,905$$

1.5 Codage arithmétique

Principe : une séquence S=(s_1,…,s_N) de N symboles (à valeurs dans un alphabet fini A) est traduite par un intervalle de [0,1[dont la longueur est égale à la probabilité de la séquence. Cet intervalle est codé par un nombre en binaire dont la longueur (en nombre de bits) est égale à $\lceil \log_2(1/p(S)) \rceil$ ($\lceil\ \rceil$ signifie entier juste supérieur à). On montre que, pour une source i.i.d, ce codage est optimal (équivalent à un code de Huffman déterminé pour une source N-étendue).

1.5.1 Notations

Alphabet de la source : $A = \{a_1 a_2 ... a_M\}$

Séquence : $S = \{s_1 s_2 ... s_N\}$.

On suppose la source i.i.d de distribution de probabilité : $p_m = p(s_n = a_m)$ pour m=1 :M.

On définit la distribution cumulée : $c_m = \sum_{i=1}^{m-1} p_i$ pour m=1 :M+1 avec $c_1 = 0$ et $c_{M+1} = 1$.

1.5.2 Algorithme de codage

Il consiste à créer une succession d'intervalles emboîtés de la forme $\phi_k(S) = |\alpha_k, l_k\rangle$ où α_k désigne la borne inférieure de l'intervalle et l_k sa longueur.

- Conditions initiales :

$\phi_0(S) = |0,1\rangle$

- Pour k=1 :N (tant qu'il reste des symboles s_k à transmettre
  - On détermine m tel que $s_k = a_m$
  - $\phi_k(S) = |\alpha_k, l_k\rangle = |\alpha_{k-1} + c_m l_{k-1}, p_m l_{k-1}\rangle$

remarque : on montre aisément que les intervalles successifs sont emboîtés les uns dans les autres.

On code $\phi_N(S) = |\alpha_N, l_N\rangle$ par un nombre (écrit en binaire) qui appartient à l'intervalle. Ce nombre (la valeur du codage) peut s'écrire :

$v = b_1 2^{-1} + b_2 2^{-2} + ... + b_{NB} 2^{-NB}$. La précision de ce nombre 2^{-NB} doit être inférieure ou égale à la longueur de l'intervalle $l_N = \prod_{k=1}^{N} p(s_k) = p(S)$.

D'où le nombre de bits du code associé à la séquence S : $NB = \lceil \log_2(1/p(S)) \rceil$

1.5.3 Algorithme de décodage

Il se fait uniquement à partir de la valeur v du code reçu. On représente la séquence décodée comme $\hat{S}(v) = \{\hat{s}_1(v)\hat{s}_2(v)...\hat{s}_N(v)\}$.

- pour k=1 :N
  - o $\hat{s}_k(v) = \{a_m : c_m \le v < c_{m+1}\}$: on détermine le symbole dont l'intervalle contient la valeur v
  - o $v = \frac{v - c_m}{p_m}$: on « annule » la contribution du symbole $\hat{s}_k(v)$ à la définition de v. On dilate l'intervalle [c_m c_{m+1}[en [0 1[

Remarque : Le décodeur doit connaître le nombre de symboles N à décoder sans quoi il continue à décoder…

1.5.4 Exemple

La source émet les symboles B (blanc) et N (noir) avec les probabilités respectives 3/4 et 1/4. On considère la séquence : BBNBN

La figure suivante représente la progression des intervalles $\phi_k(S) = |\alpha_k, l_k\rangle$ lors du codage.

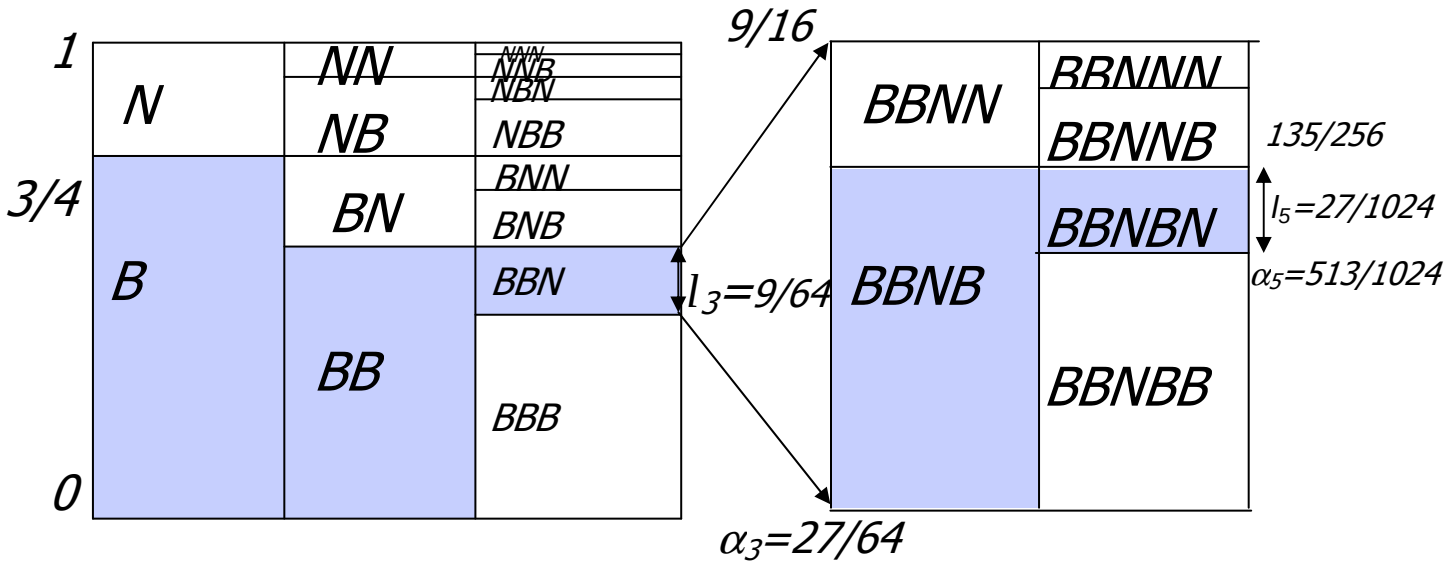


Figure : Progression de l'intervalle $\phi_k(S) = |\alpha_k, l_k\rangle$ pour coder une séquence de 5 symboles dans l'alphabet A={N,B} de probabilité p_1=3/4, p_2=1/4

Le code est un nombre binaire appartenant à l'intervalle [513/1024 135/256[. Le nombre de bits est donné par $NB = \lceil \log_2(1/l_5) \rceil = \lceil \log_2(1024/27) \rceil = 6$.

Un code possible est '1 0 0 0 0 1' de valeur $v = 33/64$.

Le décodage s'opère suivant le schéma donné Figure suivante :

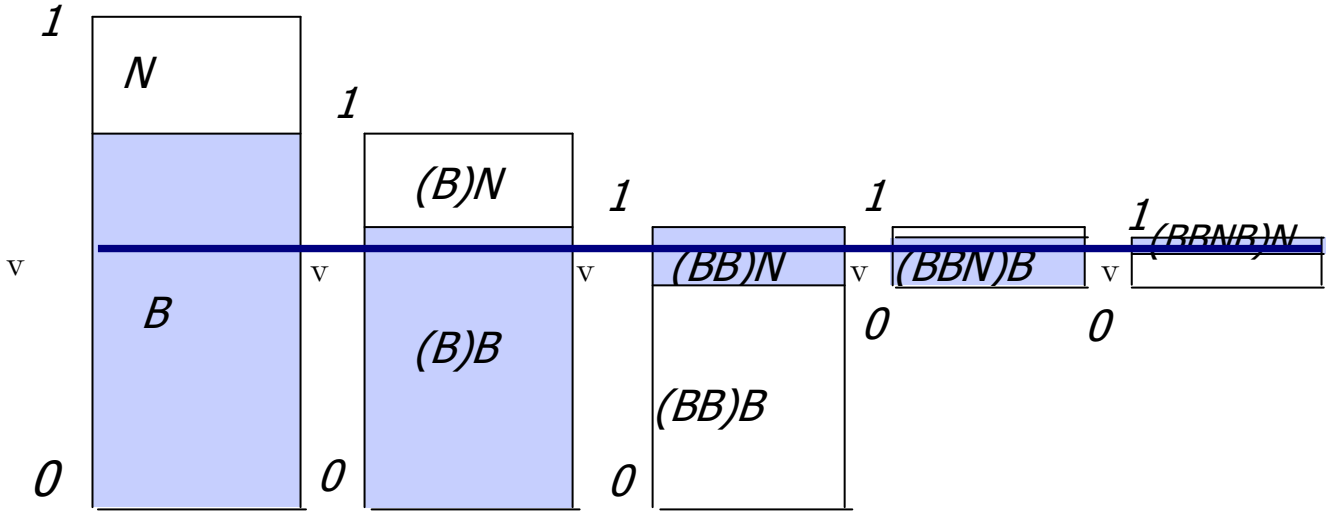


Figure : Progression de la valeur du code et des symboles décodés durant le décodage du code '1 0 0 0 0 1' sur 5 symboles pour l'alphabet A={N,B} de probabilité p_1=3/4, p_2=1/4

1.5.5 Optimalité du code *:*

Précisons qu'il faut ajouter une entête à la séquence codée qui devra comporter :

- Le nombre de symboles à décoder
- La liste des probabilités des symboles ou celle des probabilités cumulées.
- Le fait que $\lceil \log_2(1/p(S)) \rceil = \log_2(1/p(S)) + e$ avec $\varepsilon < 1$

Considérons une entête de E bits. Remarquons que E peut être limité à $\varepsilon < 1$si on transmet plusieurs séquences de mêmes caractéristiques.

On obtient pour une séquence de N symboles : $NB = \log_2(1/p(S)) + E$

Le nombre de bits par symbole vaut donc : $l = NB/N = -\sum_{k=1}^{N} \log_2(p(s_k)) + E/N$.

La moyenne statistique de cette quantité vaut donc :

$\overline{l} = H(X) + E/N$. Cette longueur moyenne tend vers l'entropie H(X) de la source à transmettre lorsque N tend vers l'infini. Si l'entête se limite à $\varepsilon < 1$, on obtient $\overline{l} < H(X) + 1/N$ qui est la limite attendue pour un code optimal opérant sur une source N-étendue.

1.5.6 Utilisation du codage arithmétique

Ce codage est en particulier utilisé dans les normes nouvelles pour le codage d'images en particulier pour la norme JPEG2000 (codage d'images fixes) et H264/AVC (codage video). Ces deux normes utilisent une version adaptative de l'algorithme, l'algorithme CABAC (codage arithmétique binaire adaptatif selon le contexte).

1.6 Conclusions sur le codage entropique – Introduction aux autres méthodes de codage (avec pertes)

On a vu que le codage dit entropique permettait un codage sans perte ($\hat{X} = X$) mais :

d'une part ce codage n'est possible que dans le cas d'une source discrète et cela signifie qu'en pratique on l'utilisera soit sur des données discrètes par nature (instructions de programme ...) soit après quantification de données « naturellement » continues,

d'autre part le taux de compression R réalisé n'est généralement pas suffisant pour la plupart des applications audio ou vidéo (typiquement 1≤R<3)

Par ailleurs pour ces dernières applications, la reconstruction exacte du signal de départ n'est pas indispensable et les critères subjectifs (qualité perceptible du son ou de l'image) seront les plus importants pour valider ou non les différentes techniques.

Nous allons donc maintenant décrire plusieurs méthodes de codage avec pertes.

La première d'entre elles, la **quantification scalaire (chapitre 2)** opère une quantification (discrétisation en amplitude) de chaque échantillon du signal à coder. Les valeurs quantifiées peuvent être dans le cas le plus simple (celui des convertisseurs analogique-numérique) réparties uniformément sur un intervalle borné : c'est la quantification uniforme, ou bien réparties de façon à minimiser la distorsion introduite par la quantification. Cette première technique n'utilise pas la corrélation qui existe entre les points successifs d'un signal ou les pixels proches d'une image.

Une façon de prendre en compte cette corrélation entre échantillons est d'opérer une transformée orthogonale (idéalement la Transformée de Karhunen Loeve) sur un vecteur constitué d'échantillons successifs avant quantification (le décodeur effectuera la transformée inverse).Ce sera le **codage par transformée (chapitre 3).**

Nous étudierons ensuite la **quantification prédictive (chapitre 4)** qui permet d'extraire la corrélation existant entre les échantillons avant de les quantifier.

Enfin, on peut directement chercher à quantifier de façon optimale un vecteur formé d'échantillons successifs ou (non exclusif) un vecteur issu d'une transformation préalable (prédiction, transformée de Fourier). C'est la **quantification vectorielle (chapitre 5)**.

Nous signalerons dans chaque cas les applications des différentes techniques pour les signaux de parole ou audio.

2. Quantification scalaire

2.1 Introduction - définition

On définit un quantificateur scalaire à L niveaux comme l'application Q :

Q : $R \rightarrow C = \{y_1, ..., y_2\}$

$x \rightarrow Q(x)$

C est l'ensemble des valeurs possibles de sortie du quantificateur, il est aussi appelé dictionnaire (ce terme sera plutôt utilisé en quantification vectorielle).
Les valeurs $\mathbf{y_i}$ sont appelées valeurs de quantification ou **représentants**.
Dans la pratique, L, le nombre de représentants est fini et est égal à 2^b de telle sorte qu'on peut, lors du codage, spécifier une valeur de sorte y_i par son numéro i ou b bits (b sera la résolution du codeur).
On peut supposer sans perte de généralité que

$y_1 < y_2 < ... < y_L$

Remarque :
Dans le cas d'un codage entropique utilisé après une quantification scalaire, on peut associer des chaînes binaires (mot de code) de tailles différentes aux L valeurs y_i en fonction de leur probabilité. On obtiendra alors une longueur moyenne des mots inférieure ou égale à b.

On associe à un quantificateur sur L représentants y_i une partition des valeurs de départ en cellules ou intervalles

$$R_i = \{x \in \Re / Q(x) = y_i\}$$

$$\bigcup R_i = \Re \quad \text{et} \quad R_i \bigcap R_j = 0 \quad \forall i \neq j$$

Une cellule qui n'est pas bornée est appelée cellule de saturation ou surcharge (overload).
Une cellule bornée est une cellule de granulation.
Le quantificateur est dit régulier si lorsque deux valeurs d'entrée a et b sont quantifiées par la même valeur y n'importe quelle valeur comprise entre a et b est aussi quantifiée par y.
Chaque cellule R_i est de la forme (x_{i-1}, x_i) avec $y_i \in (x_{i-1}, x_i)$.
Le quantificateur peut être représenté suivant le schéma :

$x_0 = -\infty$ et $x_L = +\infty$ si toute valeur d'entrée est admissible. Si les valeurs d'entrée sont bornées x_0 et x_L représentent les bornes de variation de x.

2.2 Mesure des performances d'un quantificateur

Quand une valeur d'entrée est quantifiée Q(x) = y, on introduit une erreur, l'erreur de quantification e = x - Q(x). Cette grandeur est bien sûr aléatoire puisque x l'est.
Pour mesurer la performance du quantificateur on évalue généralement la distorsion moyenne comme l'erreur quadratique moyenne (EQM) (MSE en anglais) apportée par le quantificateur.

$$D = E\left(\left(X - Q(X)\right)^2\right)$$
$$= \sum_{i=1}^{L} \int_{R_i} \left(x - y_i\right)^2 f_x(x)dx$$

Pour un quantificateur régulier

$$D = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} \left(x - y_i\right)^2 f_x(x)dx$$

L'utilisation de ce critère est simple et motivée par son sens immédiat de puissance moyenne d'un signal d'erreur.
Pourtant on sait par ailleurs que ce critère ne correspond pas à une mesure subjective de qualité (dans le domaine audio en particulier). Un critère relatif de type $Rsb = 10\log\frac{E(X^2)}{D}$ sera plus adéquat (mais non idéal car la qualité perçue auditivement dépend aussi du spectre).

Si on quantifie un signal temporel x_n réalisation d'un signal aléatoire X_n, la quantité D est à priori dépendante de l'instant n.

Si le signal est stationnaire $f_{X_n}(x) = f_X(x)$ et D est constante.

Si en plus le signal peut être considéré comme ergodique, on peut évaluer D par

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \left(x_n - Q(x_n)\right)^2$$

On peut distinguer deux types d'erreur de quantification :
l'erreur de granulation qui est l'erreur pour une entrée x appartenant à l'une des cellules bornées R_2 ... R_{L-1}
l'erreur de saturation lorsque $x<y_1$ (x_0) ou $x>y_L$ (x_L)
l'erreur de granulation est relativement petite et intervient pour presque toutes les valeurs d'entrée
l'erreur de saturation est rare mais peut avoir une très grande amplitude.

2.3 Quantification uniforme

Un quantificateur uniforme est un quantificateur régulier pour lequel les points frontières x_i sont également espacés et les représentants y_i sont au milieu des intervalles R_i. L'écart entre les frontières des cellules ou les représentants s'appelle le pas de quantification.

On a donc: $y_i - y_{i-1} = x_i - x_{i-1} =$ **Δ=pas de quantification**

et $y_i = \frac{x_{i-1}+x_i}{2}$ pour i = 1, ... L

On a $x_L - x_0 = L\,\Delta$

Remarque : En général on quantifie des signaux de moyenne nulle et on considère $-x_0 = + x_L = A$

on a alors $\Delta = \frac{2A}{L} = \frac{2A}{2^b}$ (pour une résolution b)

Le bruit de quantification s'écrit :

$$D = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x)dx + \int_{-\infty}^{x_0} (x - y_1)^2 f_X(x)dx + \int_{x_L}^{\infty} (x - y_L)^2 f_X(x)dx$$

Les deux derniers termes représentent le bruit de saturation (le premier terme est le bruit de granulation).

2.3.1 Facteur de charge

On définit le facteur de charge Γ du quantificateur par :

$\boxed{\Gamma = \frac{A}{\sigma_X}}$ ou bien si l'intervalle [x_0 x_L] n'est pas symétrique : $\Gamma = \frac{x_L}{\sigma_X}$ ou $\frac{-x_0}{\sigma_X}$

Ce facteur permet d'évaluer le risque de saturation

$$\text{proba-de-saturation} = 2\int_{A}^{\infty} f_X(x)dx = 2\int_{\Gamma\sigma_X}^{\infty} f_X(x)dx$$

exemple : pour X gaussien et $\Gamma = 2$ la probabilité de saturation = 0,045

Si on prend un facteur de charge Γ assez grand (≥ 4) la probabilité de saturation est très faible et la distorsion associée négligeable.

2.3.2 Calcul du bruit de granulation

$$D_{gran} = \sum_{i=1}^{L} \int_{R_i} (x - y_i)^2 f_X(x)dx \quad \text{avec} \quad R_i = \left(y_i - \frac{\Delta}{2}, y_i + \frac{\Delta}{2} \right)$$

on pose $x - y_i = x - Q(x) = q$ erreur de granulation.

Calculons la distorsion sur l'intervalle R_i

$$\int_{R_i} q^2 f_X(x)dx = \int_{R_i} f_X(x)dx \int_{\infty} q^2 f_{X/R_i}(x)dx$$
$$= p(x \in R_i) \int_{\infty} q^2 f_{X/R_i}(x)dx$$

$$f_{X/Ri}(x)dx = \text{proba}\left(x < X < x + dx / y_i - \frac{\Delta}{2} < X < y_i + \frac{\Delta}{2} \right)$$
$$= \text{proba}\left(x - y_i < Q < x - y_i + dx / y_i - \frac{\Delta}{2} < X < y_i + \frac{\Delta}{2} \right)$$
$$= f_{Q/Ri}(q)dq$$

donc

$$D_{gran} = \sum_{i=1}^{L} p(x \in R_i) \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q^2 f_{Q/Ri}(q)dq$$

On fait généralement la supposition que $f_{Q/Ri}(q) = \frac{1}{\Delta}$. C'est-à-dire que le bruit de granulation suit une loi uniforme sur $\left(-\frac{\Delta}{2}, \frac{\Delta}{2} \right)$(quelque soit l'intervalle Ri). Ceci n'est vraiment justifié que si X suit une loi uniforme ou si Δ est très faible (hypothèse de haute résolution).

En prenant cette hypothèse :

$$D_{gran} = \frac{1}{\Delta} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} q^2 dq = \frac{\Delta^2}{12}$$

Si on exprime cette quantité en fonction de Γ et b

$$\boxed{D_{gran} = \frac{1}{3} \sigma_x^2 \Gamma^2 2^{-2b}}$$

La distorsion liée à l'erreur de granulation est proportionnelle :

- au facteur de charge(au carré) : lorsqu' on limite les risques de saturation on augmente l'erreur de granulation.
- à la variance du signal
- à 2^{-2b} : quand on augmente la résolution on diminue la distorsion

2.3.3 Rapport signal sur bruit

$$RSB_{gran} = 10 \log \frac{\sigma_x^2}{D_{gran}}$$

$$\boxed{RSB_{gran} = 6{,}02b + 4{,}77 - 20 \log \Gamma}$$

Le rapport signal sur bruit décroît lorsque le facteur de charge croît (en négligeant la distorsion de saturation) : (pour $\Gamma = 4$ $\quad 20 \log \Gamma = 12$ dB)

Par ailleurs le RSB_{gran} augmente de 6 dB pour une augmentation de 1 bit dans la résolution du quantificateur.

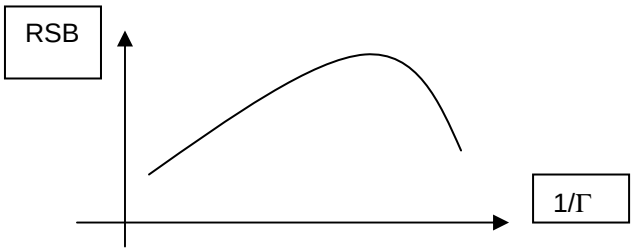
Remarque 1 :Si le signal x suit une loi uniforme : $\sigma_x = \frac{A}{\sqrt{3}} \Rightarrow \Gamma = \sqrt{3}$

RSB = 6,02 b

La quantification uniforme est alors parfaitement adaptée.

Remarque 2 :Si le facteur de charge n'est pas suffisamment grand, on ne peut plus négliger le bruit de saturation.

Si on trace le RSB (en tenant compte de la saturation) en fonction de $\beta = 1/\Gamma$, on obtient une courbe qui a l'allure suivante :



On peut aussi voir cette courbe comme l'évolution du RSB avec la puissance d'entrée du signal σ_x pour un quantificateur donné (avec y_i fixés).
On voit alors que le quantificateur fonctionne de manière optimum pour une puissance donnée du signal d'entrée et qu'il se comporte moins bien lorsqu'on s'écarte de cette valeur. En d'autres termes, les performances d'un quantificateur sont très sensibles au niveau du signal d'entrée.
Des signaux tels que la parole ou la musique sont non stationnaires et leur puissance à court terme varie fortement dans le temps (la dynamique de σ est d'environ 30 dB pour la parole et 60 dB ou plus pour la musique). Le nombre de bits

de résolution requis pour ce type de signaux dépend du RSB minimum que l'on veut obtenir.
Par exemple, pour une qualité téléphonique, il faut 12 bits pour que le RSB soit supérieur à 25 dB pour des signaux de puissance égale à celle où commence la saturation -30 dB.

2.4 Conception d'un quantificateur scalaire optimal.

On suppose que l'on cherche à quantifier de façon optimale sur L niveaux une réalisation d'un processus aléatoire stationnaire de loi $f_X(x)$ connue (ou estimable).

Intuitivement, la longueur des intervalles $[x_{i-1}, x_i]$ sera d'autant plus petite que $f_X(x)$ sera importante sur ces intervalles.

On rappelle que la distorsion est donnée par :

$$D = \sum_{i=1}^{L} \int_{x \in Ri} (x - y_i)^2 f_X(x) dx = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x) dx$$

Pour définir un quantificateur il faut trouver la partition $\{R_1 \ldots R_L\}$ et les représentants $(y_1 \ldots y_L)$ qui minimisent D.
Ce problème n'admet pas de solution simple. Il n'existe que deux conditions nécessaires d'optimalité. Si on connaît les représentants, on peut trouver la meilleure partition et si on se donne la partition on peut déduire les représentants.

2.4.1 Conditions d'optimalité

1) Etant donné le dictionnaire $\{y_1 \ldots y_L\}$, on cherche les x_i (c'est à dire la partition $\{R_i\}$) tels que

$$\frac{\partial D}{\partial x_i} = 0$$

$$\Leftrightarrow \frac{\partial}{\partial x_i}\left(\int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x) dx + \int_{x_i}^{x_{i+1}} (x - y_{i+1}) f_X(x) dx \right) = 0$$

$$\Leftrightarrow (x_i - y_i)^2 f_X(x_i) - (x_i - y_{i+1})^2 f_X(x_i) = 0$$

$$\Leftrightarrow x_i - y_i = -(x_i - y_{i+1})$$

$$\Leftrightarrow \boxed{x_i = \frac{y_i + y_{i+1}}{2}}$$ la frontière est en milieu des représentants.

Ceci correspond à la **règle du plus proche voisin (PPV).**

$$R_i = \left\{ x / (x - y_i)^2 \leq (x - y_j)^2 \, \forall j \in \{1 \ldots L\} \right\}$$

2) Etant donnée une partition $\{R_i\}$ on cherche les représentants y_i qui minimisent D.

$$\frac{\partial D}{\partial y_i} = 0$$

$$\Leftrightarrow \frac{\partial}{\partial y_i} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x) dx = 0$$

$$\Leftrightarrow \int_{x_{i-1}}^{x_i} 2(x - y_i) f_X(x) dx = 0$$

$$\Leftrightarrow \boxed{y_i = \frac{\int_{x_{i-1}}^{x_i} x f_X(x) dx}{\int_{x_{i-1}}^{x_i} f_X(x) dx}}$$

or $$\int_{x_{i-1}}^{x_i} x f_X(x) dx = \int_{x \in R_i} f_X(x) dx \int_{-\infty}^{+\infty} x f_{X/R_i}(x) dx$$

$$\Leftrightarrow y_i = \int_{-\infty}^{+\infty} x f_{X/R_i}(x) dx$$

$= E(X / X \in R_i)$ valeur moyenne de x dans l'intervalle $]x_{i-1}, x_i]$

y_i est le centroïde de la classe R_i (la moyenne des éléments de cette classe)

2.4.2 Algorithme de Lloyd - Max

Dans la pratique on ne connaît généralement pas $f_X(x)$. Pour construire le quantificateur, on utilise une base d'apprentissage composée d'un grand nombre N >> L d'échantillons représentatifs de la source que l'on veut quantifier.
L'algorithme de Lloyd Max est un algorithme itératif qui vérifie alternativement les deux conditions d'optimalité .

1) On initialise le dictionnaire $(y_1, ..., y_2)$ de façon aléatoire (ou uniforme)

2) Connaissant le dictionnaire, on classe tous les échantillons de la base en leur attribuant le numéro de leur plus proche représentant. (On détermine ainsi implicitement les R_i).

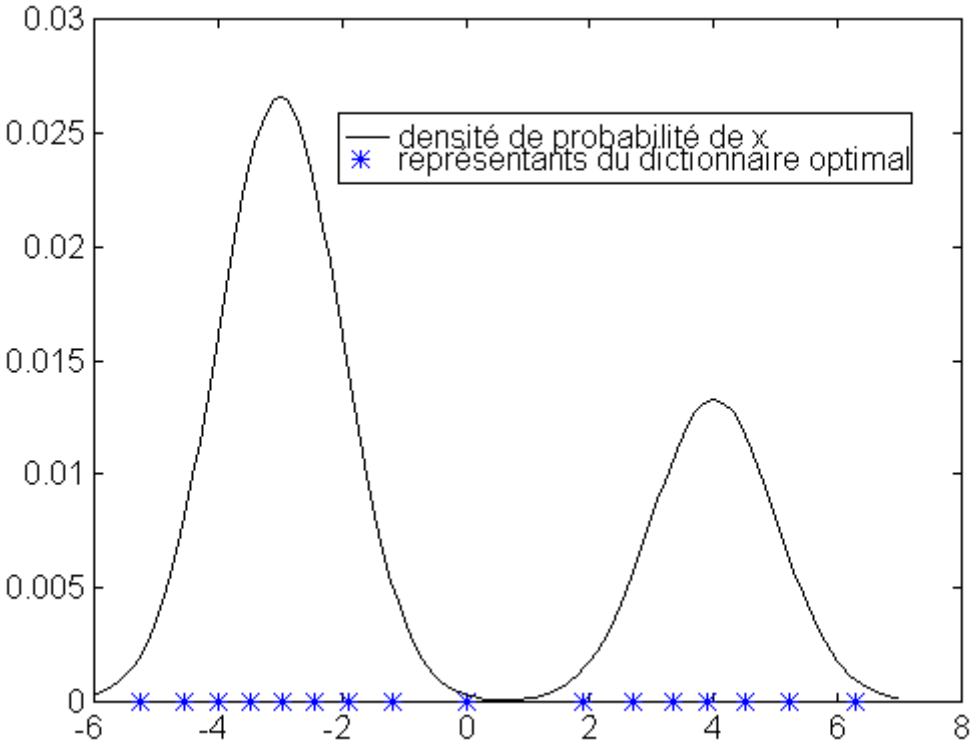
$R_i = \{ x_n / Q(x_n) = y_i \}$

3) A partir de ces classes on calcule de nouveaux représentants qui sont les moyennes des échantillons appartenant à une classe.

4) On calcule la distorsion moyenne associée à cette base et on arrête l'algorithme si la distorsion ne décroît plus de manière significative sinon on reprend les deux étapes précédentes.

On ne tend pas forcément vers le minimum global mais simplement vers un minimum local par cette technique.

Exemple :
Soit x un échantillon d'un signal dont la densité de probabilité est la somme pondérée de deux densités gaussiennes d'écart type unité et de moyennes respectives –3 et 4. La densité de probabilité de X est représentée sur la figure suivante ainsi que le quantificateur (obtenu par l'algorithme de Lloyd Max).



2.4.3 Quantification haute résolution (b >=8) - cas général

Si on se place dans le cas : L grand, $f_X(x)$ est raisonnablement « lisse », la distorsion donnée par :

$$D = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x) dx$$ peut être approximée en considérant

$f_X(x) = f_X(y_i) \quad \text{sur} \quad [x_{i-1}, x_i]$

$$y_i = \frac{x_i + x_{i-1}}{2}$$

$$D = \sum_{i=1}^{L} f_X(y_i) \int_{x_{i-1}}^{x_i} \left(x - \frac{x_i + x_{i-1}}{2} \right)^2 dx$$

$$= \sum_{i=1}^{L} f_X(y_i) \int_{-\frac{\Delta_i}{2}}^{\frac{\Delta_i}{2}} y^2 dy \qquad \text{avec } \Delta_i = x_i - x_{i-1}$$

$$\boxed{D = \sum_{i=1}^{L} \frac{\Delta_i^3}{12} f_X(y_i)}$$

(Si on considère $P_i = \text{proba}\left(x \in [x_{i-1}, x_i]\right) = f_X(y_i)\Delta_i$
on peut encore écrire

$$\boxed{D = \sum_{i=1}^{L} P_i \frac{\Delta_i^2}{12} = E\left(\frac{\Delta^2}{12}\right)} \qquad)$$

Notons $\alpha_i^3 = f_X(y_i)\Delta_i^3$

comme $\sum_{i=1}^{L} \alpha_i = \sum_{i=1}^{L} \left(f_X(y_i)\right)^{1/3} \Delta_i \approx \int_{-\infty}^{+\infty} f_X(x)^{1/3} dx = cste$

On cherche à minimiser $D = \sum_i \alpha_i^3$ sous la contrainte $\sum_i \alpha_i = cste$

$$\rightarrow \frac{\partial}{\alpha_i}\left(\sum \alpha_i^3 - \lambda \sum \alpha_i\right) = 0 \Leftrightarrow 3\alpha_i^2 = \lambda \Leftrightarrow \alpha_i = \alpha$$

Ce qui implique

$$\boxed{f_X(y_1)\Delta_1^3 = ... = f_X(y_2)\Delta_L^3}$$

Ce qui signifie que l'intervalle Δ sera d'autant plus petit que la probabilité d'avoir une donnée dans cet intervalle sera grande. ($\underline{\Delta_i = cste \text{ pour une loi uniforme}}$) .

D'autre part tous les intervalles auront la même contribution à la distorsion qui s'écrit

$$D = \frac{L\alpha^3}{12} \qquad \text{avec} \quad \alpha = \frac{1}{L} \int_{-\infty}^{+\infty} \left(f_X(x)\right)^{1/3} dx$$

$D = \frac{1}{12L^2}\left[\int \left(f_X(x)\right)^{1/3} dx\right]^3$ démonstration non rigoureuse

Pour une source gaussienne centrée de variance σ_X^2

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-x^2/2\sigma_X^2}$$

$$D = \frac{1}{12\ L^2}\left[\int \frac{1}{\left(2\Pi\ \sigma_X^2\right)^{\frac{1}{6}}} e^{-\frac{x^2}{6\sigma_X^2}dx}\right]^3 = \frac{1}{12\ L^2}\left[\frac{\left(2\Pi\ \sigma_X^2\right)^{1/2}\sqrt{3}}{\left(2\Pi\ \sigma_X^2\right)^{1/6}}\int \frac{1}{\sqrt{2\pi}\sqrt{3}\sigma_X} e^{-\frac{x^2}{2\left(\sqrt{3}\ \sigma_X\right)^2}dx}\right]^3$$

$$D = \frac{1}{12L^2}\left[\sqrt{3}\left(2\pi\sigma_X^2\right)^{1/3}\right]^3$$

$$\boxed{D = \frac{\sqrt{3}}{2}\Pi\sigma_X^2 2^{-2b}}$$

Pour d'autres lois on aura :

$\boxed{D = C\sigma_X^2 2^{-2b}}$ avec C une cste qui dépend de la loi de X.

La distorsion est proportionnelle :

- à la variance du signal
- à 2^{-2b} : quand on augmente la résolution on diminue la distorsion

2.5 Quantification logarithmique

2.5.1 Principe

Pour le quantificateur uniforme

On a vu que $D_{gran} = \frac{\Delta^2}{12}$ avec $\Delta\ cst.$

$\Rightarrow RSB = 10\log\frac{\sigma^2}{\Delta^2}12$ dépend de σ

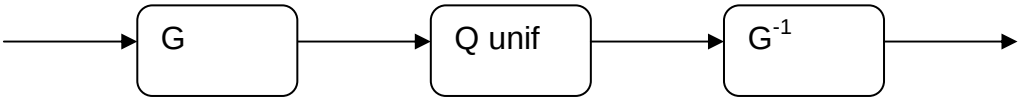
L'idée est de rendre le pas de quantification adapté à la puissance du signal pour avoir un RSB à peu près constant pour une large gamme de puissance du signal d'entrée.

Il faut alors : $\left|\frac{\Delta_i}{y_i}\right| = \frac{\left|x_i - x_{i-1}\right|}{\left|y_i\right|} \approx cste.$

On peut alors réaliser ceci en trois étapes, à savoir :

1) Une transformation non linéaire sur x : G

2) Une quantification uniforme sur le résultat : $\Delta = G(x_i) - G(x_{i-1})$
3) La transformation inverse sur le résultat quantifié : G^{-1}.



Dans l'hypothèse haute résolution :

$$G'(y_i) \cong \frac{G(x_i) - G(x_{i-1})}{x_i - x_{i-1}} = \frac{\Delta}{\Delta_i}$$
$$\rightarrow G'(y_i) = \frac{\Delta}{cste}\frac{1}{y_i} = \frac{\alpha}{y_i}$$
$$\Rightarrow G(x) = \alpha \log x$$

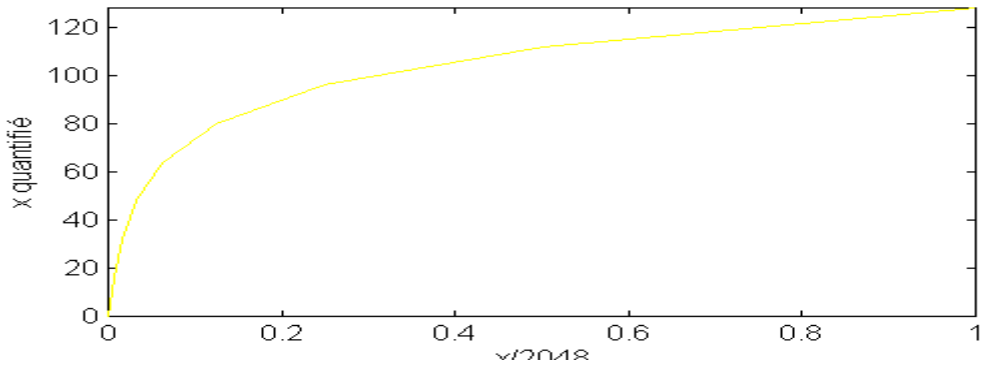
2.5.2 Norme G711 (de l'Union Internationale des Télécommunications) : codage MIC (PCM) à 64 kbits/s.

Le signal de parole est échantillonné à 8 kHz
Il est quantifié sur 8 bits suivant la loi A : (A = 87,56)

$$G(x) = \frac{A|x|}{1+\ln A} sgn(x) \quad \text{pour} \quad \frac{|x|}{x_{\max}} \le \frac{1}{A}$$

$$G(x) = x_{\max} \frac{1+\ln A\frac{|x|}{x_{\max}}}{1+\ln A} \quad \text{pour} \quad \frac{1}{A} < \frac{|x|}{x_{\max}} \le 1$$

Cette transformation est approximée par une courbe à 8 segments et le signal est préalablement quantifié sur 12 bits (-2048 → 2047) .
1 bit de signe
3 bits pour le segment
4bits pour la valeur dans le segment



3. Allocation optimale des bits et codage par transformée

3.1 Allocation optimale des bits

On se pose le problème d'allocation de ressources dans les termes suivants :
On dispose d'un nombre fini de bits à répartir sur un ensemble de paramètres à coder (une trame de parole, la matrice des pixels d'une image) et on recherche la meilleure répartition (celle qui donnera la meilleure qualité objective ou subjective).
Dans les codeurs traditionnels chaque échantillon est quantifié avec le même nombre de bits puisque le plus souvent ces échantillons ont tous la même « importance » (la même variance).
Nous allons résoudre le problème d'allocation optimale des bits sur des échantillons ou paramètres de variances différentes puis nous verrons comment il est possible de transformer un vecteur d'échantillons de même variance pour obtenir un vecteur de paramètres de variances différentes sur lesquels on peut faire une allocation optimale des bits.

3.1.1 Position du problème

Supposons que nous ayons un ensemble de k v.a. X_1, X_2, …, X_k de moyenne nulle et de variance $\sigma_i^2 = E(X_i^2) \quad i = 1,...,k$
Si on considère la densité de probabilité des X_i connue, on peut concevoir un quantificateur optimal pour chaque X_i pour n'importe quel nombre de niveaux de quantification L_i. Il est souvent souhaitable d'avoir $L_i = 2^{bi}$ pour pouvoir coder X_i sur $b_i = \log_2 L_i$ bits.
Il suffit en fait d'avoir :
$\Pi L_i = 2^b$ pour coder sur b bits les k v.a. conjointement.
Supposons que l'on note W_i (b_i) la distorsion (au sens de l'erreur de quantification quadratique moyenne) obtenue en quantifiant de façon optimale X_i sur b_i bits.
La distorsion totale s'écrit donc :

$$D = \sum_{i=1}^{k} W_i(b_i)$$

Le problème d'allocation peut se formuler ainsi :
Trouver les b_i i = 1, 2, … k qui minimisent

$$D(b) = \sum_{i=1}^{k} W_i(b_i) \qquad b = (b_1, b_2, ..., b_k)$$

sous la contrainte : $\sum_{i=1}^{k} b_i \le B$ (B : nombre de bits disponibles)

Il n'est pas évident de trouver un algorithme exact qui résolve ce problème dans le cas général :
On peut faire la recherche sur toutes les combinaisons possibles (avec les b_i entiers et satisfaisant $\sum b_i \le B$) mais le nombre de calculs à effectuer est prohibitif et il est difficile de disposer des distorsions $W_i(b_i)$ pour toutes les valeurs de b_i testées.

En revanche, en faisant l'hypothèse de haute résolution, on peut obtenir une solution exacte, optimale (mais avec des b_i non forcément entiers).

3.1.2 Allocation optimale dans l'hypothèse haute résolution

Dans ce cas $\boxed{W_i(b_i) = h_i \sigma_i^2 2^{-2b_i}}$ avec h_i une constante qui dépend de la densité de probabilité $f_i(x)$ de la variable normalisée $\frac{X_i}{\sigma_i}$

$$h_i = \frac{1}{12}\left(\int_{-\infty}^{+\infty} (f_i(x))^{\frac{1}{3}} dx\right)^3$$

On rappelle que pour une loi gaussienne $h_g = \frac{\sqrt{3}\Pi}{2}$

Remarque : La formule $W_i(b_i) = h_i \sigma_i^2 2^{-2b_i}$ n'est à priori pas correcte pour b_i faible (1 bit ou 2 bits) mais se révèle satisfaisante pour des valeurs plus élevées (4 bits→ 16 bits).

Le problème de minimisation de $D = \sum_i h_i \sigma_i^2 2^{-2b_i}$ sous la contrainte $\sum b_i = B$ est résolue en utilisant la méthode des multiplicateurs de Lagrange :
On annule la dérivée par rapport à b_i de $D + \lambda \sum_i b_i$

Développement :

$$h_i \sigma_i^2 (-2\log 2) 2^{-2b_i} + \lambda = 0$$

On calcule λ en écrivant la contrainte: $\sum b_i = B$
Après quelques lignes de calculs on obtient :

$$b_i = \frac{1}{2}\log_2\left(2^{2B}\right)^{1/k} \frac{h_i}{(\Pi h_i)^{1/k}} \frac{\sigma_i^2}{\rho^2}$$

$$\boxed{\begin{array}{l} b_i = \dfrac{B}{k} + \dfrac{1}{2}\log_2 \dfrac{h_i}{H} + \dfrac{1}{2}\log_2 \dfrac{\sigma_i^2}{\rho^2} \\ \text{avec } \rho^2 = \text{moyenne géométrique des } \sigma_i^2 \\ \text{et } \quad H = \text{moyenne géométrique des } h_i \end{array}}$$

$\Leftrightarrow$

La distorsion totale vaut alors :

$$D = \sum_{i=1}^{k} h_i \sigma_i^2 \, 2^{-2B/k} \, \frac{H}{h_i} \frac{\rho^2}{\sigma_i^2}$$

$$\Leftrightarrow \quad \boxed{D = k\, H\, \rho^2 \, 2^{-2\frac{B}{k}}}$$

Chaque quantificateur a la même contribution à la distorsion D, cette contribution est : $H\, \rho^2 \, 2^{-2\frac{B}{k}}$

Ce résultat est équivalent à celui qu'on obtiendrait pour un ensemble de k v.a. de même variance ρ^2, de même loi de constante H, quantifiées chacune sur $\bar{b} = \dfrac{B}{k}$ bits.

L'équation $b_i = \bar{b} + \dfrac{1}{2}\log_2 \dfrac{h_i}{H} + \dfrac{1}{2}\log_2 \dfrac{\sigma_i^2}{\rho^2}$ montre qu'il faut allouer plus de bits aux composantes de plus grande puissance (variance).

Remarque :cette solution « optimale » présente certains inconvénients :
$\sum b_i = B$ est entier mais les b_i ne le sont pas forcément, certains b_i peuvent même être négatifs.
En pratique, on traite ces cas en remplaçant les valeurs négatives par des zéros et les $b_i > 0$ par leur valeur entière la plus proche mais on peut alors perdre $\sum b_i = B$.
Une alternative peut être trouvée en optimisant localement la distorsion en testant les jeux de b_i entiers satisfaisant $\sum b_i = B$ autour de la valeur optimale.
On peut aussi employer une technique d'allocation autre qui impose la contrainte b_i entier.
Exposons maintenant un algorithme simple qui correspond à cet objectif.

3.1.3 Allocation de bits (b_i entier) par l'algorithme du « Greedy »

Cet algorithme n'est pas optimal mais il alloue les bits par incrément (1 bit à chaque itération) d'une façon très satisfaisante (il est utilisé dans le codeur MPEG). L'algorithme du Greedy est ainsi nommé car il agit à court terme, cherchant à obtenir une satisfaction (la réduction de la distorsion) immédiate sans considérations pour l'effet à long terme (distorsion finale globale) de ce choix.

Algorithme du Greedy

On note :

- $b_i(m)$ le nombre total de bits alloués au $i^{ème}$ quantificateur après l'itération m (quand m bits ont été partagés entre les X_i).
- $W_i(m)$: la « demande » associée au $i^{ème}$ quantificateur (la distorsion pour ce quantificateur après l'itération m).

étape 0 : $b_i(0) = 0 \quad \forall i$
m= 0 $W_i(0) = E(X_i^2)$

étape 1 : on trouve l'indice j qui maximise la demande
$j = \arg\max W_i(m)$

étape 2 : on alloue un bit au $j^{ième}$ quantificateur
$b_j (m+1) = b_j(m) + 1$
$b_i (m+1) = b_i(m) \quad \forall i \neq j)$
les demandes sont actualisées
$W_j(m+1) = W_j(m) / \theta$
Avec $1,4 < \theta < 2,3$ ($\theta = 2$ dans le cas haute résolution)

étape 3 : si m < B-1
m = m+1 et on reprend l'étape 1.

3.2 Codage par Transformée

3.2.1 Principe.

Supposons que l'on ait un bloc d'échantillons successifs (vecteur) d'un processus aléatoire stationnaire à coder avec un nombre fixé de bits.
Appelons X ce vecteur aléatoire $X = (X_1, X_2, \ldots, X_k)^T$
Ces échantillons qui d'après l'hypothèse de stationnarité ont la même variance peuvent présenter une corrélation importante. Cette corrélation entraîne une certaine redondance qui est conservée dans les échantillons quantifiés.
L'idée du codage par transformée est qu'en opérant une transformation linéaire sur X on peut obtenir un nouveau vecteur Y dont les composantes sont moins corrélées que celles de X et que l'information peut y être plus compacte, c'est-à-dire concentrée sur quelques composantes, au lieu d'être uniformément répartie sur toutes les composantes. On espère alors quantifier ces composantes de façon plus efficace que pour X.

On effectuera au moment du codage la transformée Y=TX.
On quantifiera scalairement les composantes du vecteur transformé Y afin d'obtenir le vecteur $\hat{Y}$ (quantifié).
Au décodage le vecteur $\hat{X}$ sera obtenu par transformation inverse du vecteur $\hat{Y}$ (quantifié) .
Le principe du codage par Transformée est résumé dans le schéma suivant:

$$X \begin{vmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{vmatrix} \rightarrow (T) \rightarrow Y \begin{vmatrix} Y_1 \rightarrow [Q_1] \rightarrow \\ Y_2 \rightarrow [Q_2] \rightarrow \\ \vdots \\ Y_k \rightarrow [Q_k] \rightarrow \end{vmatrix} \hat{Y} \begin{vmatrix} \hat{Y}_1 \\ \hat{Y}_2 \\ \vdots \\ \hat{Y}_k \end{vmatrix} \rightarrow (T^{-1}) \rightarrow \hat{X} \begin{vmatrix} \hat{X}_1 \\ \hat{X}_2 \\ \vdots \\ \hat{X}_k \end{vmatrix}$$

Il y a aussi une raison « subjective » pour utiliser une transformée qui est de se référer aux outils perceptifs humains qui interviennent pour la vue ou d'audition.
L'oreille en particulier, opère au niveau de la membrane basilaire (dans l'oreille interne) une transformation du signal temporel acoustique en influx nerveux répartis suivant une échelle fréquentielle sur les fibres nerveuses du nerf auditif. Ces fibres se comportent, en première approximation et partiellement, comme un banc de filtres dont les fréquences centrales sont disposées sur une échelle pseudo-logarithmique.

Les transformées basées sur la transformée de Fourier ou la transformée en Cosinus sont très largement utilisées dans les applications de codage audio ou image. Elles permettent de différencier les RSB dans différentes bandes de fréquence et d'allouer de façon optimale les bits disponibles dans le domaine transformé (fréquentiel dans ce cas).

3.2.2 Transformées orthogonales.

La mesure de la distorsion pour un codage par transformée est :

$$Dtc = \sum_{i=1}^{k} E\left((X_i - \hat{X}_i)^2\right)$$
$$= E\left((X - \hat{X})^T (X - \hat{X})\right)$$

Notons que $\hat{X}$ est obtenu par $T^{-1}\hat{Y}$ et que l'erreur de reconstruction $X - \hat{X}$ dépend à la fois de l'erreur de quantification $Y - \hat{Y}$ et de T^{-1} qui risque d'accentuer les erreurs (si le déterminant de T^{-1} est supérieur à 1).
On impose donc à la transformation *T* d'être orthogonale c'est-à-dire telle que $T^T = T^{-1}$ pour que la norme d'un vecteur soit préservée par la transformation.

3.2.2.1 Propriétés pour les transformations orthogonales

a) $E(X^T X) = E(Y^T T T^T Y) = E(Y^T Y)$ norme conservée

b) $E\left((X - \hat{X})^T (X - \hat{X})\right) = E\left((Y - \hat{Y})^T (Y - \hat{Y})\right)$ distorsion conservée

c) $R_Y = E(YY^T) = TE(XX^T)T^T = TR_X T^T$

et puisque det(T) = 1 $\det(R_X) = \det(R_Y)$

d) Si on appelle V_i^T les lignes de T

$T^T = (V_1 V_2 \ldots V_k)$

$TT^T = I \quad \Leftrightarrow \quad \boxed{V_i^T V_j = \delta_{ij}}$ les vecteurs V_i sont orthogonaux

$Y = TX \quad \Leftrightarrow \quad Y_i = V_i^T X \quad \forall i = 1 \ldots k$

$\Rightarrow \sigma_{Yi}^2 = E(Y_i^2) = V_i^T R_X V_i$ distorsion sur Y_i

3.2.3 Transformation optimale- Transformée de Karhunen Loeve

On veut trouver la transformation orthogonale T (les Vi) telle que
$D_{tc} = \sum_{i=1}^{k} E\left((Y_i - \hat{Y}_i)^2\right)$ soit minimale avec les contraintes :

- $V_i^t V_i = 1 \quad \text{pour} \quad i = 1 \ldots k$
- $E(X^T X) = E(Y^T Y)$
  $\Leftrightarrow \sum_i \sigma_{Y_i}^2 = k\sigma_X^2$

On peut résoudre ce problème en faisant l'hypothèse de la haute résolution.

On a alors

$$D_{tc}=\sum_{i=1}^{k} h_i\sigma_{Y_i}^2 2^{-2b_i}$$
$$=\sum_{i=1}^{k} h_i V_i^T R_X V_i 2^{-2b_i}$$

On annule la dérivée par rapport aux V_i du critère à minimiser avec contraintes :

$$\sum_{i=1}^{k} h_i V_i^T R_X V_i 2^{-2b_i} - \lambda\left(\sum_i V_i^T R_X V_i - k\sigma_X^2\right) - \sum_i \mu_i (V_i^T V_i - 1)$$

d'où : $h_i R_X V_i 2^{-2bi} = \lambda R_X V_i + \mu_i V_i$

$$\Leftrightarrow \qquad \boxed{R_X V_i = \lambda_i V_i} \quad \text{avec} \quad \lambda_i = \left(\frac{\mu_i}{h_i 2^{-2bi} - \lambda}\right)$$

On trouve que les V_i (colonnes de T) doivent être les vecteurs propres de la matrice de covariance R_X.
On a alors :
$R_y=TR_XT^T=\text{diag}(\lambda_i)$

Les composantes Y_i sont décorrélées
Leur variance $\sigma_{Y_i}^2$ est égale à la valeur propre λ_i.

Cette transformation optimale qui décorrèle les composantes est la <u>transformée de Karhunen Loeve.</u>

La transformée T_{KL} optimale est donc formée de lignes qui sont les vecteurs propres (transposés) de la matrice de covariance du signal X.

Pour trouver ensuite la meilleure allocation de bits dans le domaine transformé on est ramené au problème de minimisation de : $\sum_{i=1}^{k} h_i \lambda_i 2^{-2bi}$ qui a comme solution :

$$bi = \frac{B}{k} + \frac{1}{2}\log_2 \frac{h_i}{H} + \frac{1}{2}\log_2 \frac{\lambda_i}{\left(\prod_i \lambda_i\right)^{1/k}}$$

avec la distorsion : $\boxed{D_{TKL} = kH(\Pi\lambda_i)^{1/k} 2^{-\frac{2B}{k}}}$

On pourra, pour avoir une solution entière et ne pas faire l'hypothèse haute résolution, utiliser l'algorithme du Greedy en partant des conditions initiales $W_i = \lambda_i$.

3.2.4 Gain de performance par le codage par transformée

Si on se place dans le cas d'un vecteur X composé de k échantillons d'un processus gaussien faiblement stationnaire :
$E(X_i^2) = \sigma_X^2$

On peut comparer les distorsions obtenues en quantifiant scalairement les composantes de X avec $\frac{B}{k}$ bits chacune et en quantifiant les composantes de Y sur b_i bits ($\Sigma b_i = B$)

$D_{Q-X} = k\, h_g\, 2^{-2B/k} \sigma_X^2$

$D_{TKL} = k\, h_g\, (\Pi \lambda i)^{1/k} 2^{-2B/k}$

Le gain de codage est donc

$$G_{KL} = \frac{\sigma_X^2}{(\Pi \lambda i)^{1/k}} = \frac{\frac{\sum \lambda i}{k}}{(\Pi \lambda i)^{1/k}}$$

Le gain est donc le rapport entre la moyenne arithmétique et la moyenne géométrique des valeurs propres de la matrice de covariance.
Le gain est d'autant plus important que les valeurs propres sont dispersées $\left(\frac{\lambda \max}{\lambda \min} \text{ grand}\right)$
ce qui est lié à des fortes corrélations dans le vecteur de départ X.

En pratique il arrive que certaines composantes de Y aient une variance λ_i suffisamment faible pour qu'on puisse ne pas leur allouer de bits du tout.
La Transformée a alors concentré l'information sur un nombre réduit de composantes.

3.2.5 Autres transformées

Les transformées de Fourier (TFD) et transformée en cosinus (DCT) sont souvent mises en œuvre en pratique à la place de la transformée de Karhunen Loeve (TKL) car celle-ci présente une grande complexité de calcul pour des signaux qui ne sont pas véritablement stationnaires et pour lesquels il faut actualiser la matrice de covariance et donc la matrice de transformation régulièrement.
La DCT est principalement utilisée dans les normes de codage d'images (JPEG et MPEG) et n'est pas présentée dans le cadre de ce cours.

Montrons simplement ici que la Transformée de Fourier approxime la TKL pour un signal stationnaire continu. Soit x(t) un signal stationnaire et X(f) sa TF(aléatoire elle aussi).

$r_{xx}(\tau) = E\left(x(t)x(t+\tau)\right)$ le signal est à priori corrélé avec lui–même

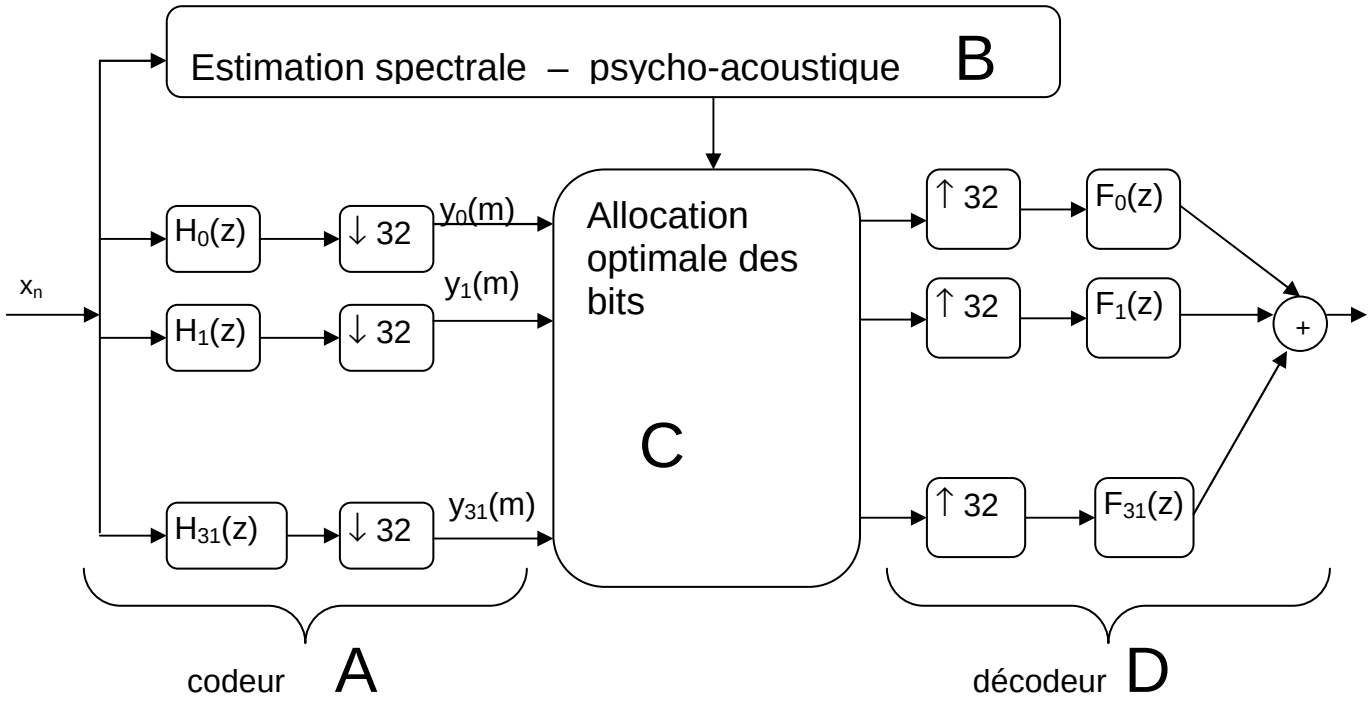
$$\begin{aligned} R_{XX}(\upsilon) &= E\left(X(f)X^*(f+\upsilon)\right) \\ &= E\left(\int x(t)\, e^{-j2\pi ft} dt \int x(u)\, e^{+j2\pi(f+\upsilon)u} du\right) \\ &= \iint r_{XX}(t-u)\, e^{-j2\pi f(t-u)}\, e^{j2\pi\upsilon u} dt\, du \\ &= P_{XX}(f)\,\delta(\upsilon) \end{aligned}$$

$X(f)$ et $X^*(f+\upsilon)$ sont décorrélés pour $\upsilon \neq 0$

3.3 Application : Codage MPEG audio

3.3.1 Principe : codeur en sous-bandes avec allocation optimale des bits dans chaque sous bande.

Le schéma de principe du codeur est le suivant :



La norme MPEG audio est séparé en 3 niveaux de compression appelés couches I II III
La norme est compatible avec le MIC, le CD, le DAT… et admet donc plusieurs fréquences d'échantillonnage possibles (16 KHz, 22.05 KHz, 32 KHz, 44.1 KHz, 48 KHz).
La couche I (succinctement décrite ci-après) permet un codage transparent (qualité non dégradée) à 192 kbps.
En sortie du codeur les données sont organisées en trames. Le débit est compris entre 8 kbps et 448 kbps.

Le principe du codeur MPEG est d'allouer dynamiquement (tous les 12 échantillons Y(m)) les bits sur les composantes de Y(m) en fonction de critères perceptifs. On ne code que les composantes audibles du signal sonore (on maximise le rapport signal sur masque).

Description succincte des différents blocs du schéma.

Bloc A :

Le signal est filtré par un banc de 32 filtres qui sont des versions décalées (en fréquence) d'un filtre FIR passe bas à 512 coefficients qui coupe à Fe/64. La sortie de ces filtres peut être alors décimée par 32 c'est-à-dire échantillonnée à la fréquence de Nyquist (2 fois la largeur de bande du filtre).

L'ensemble de ces deux opérations (filtrage, décimation) peut s'effectuer par Transformée Y = T X

T est le produit d'une matrice de DCT avec une matrice diagonale qui correspond à la pondération du signal par une courbe de type sinc (prototype passe-bas du banc de filtre).

X est un vecteur de 512 échantillons du signal d'entrée. On l'acquiert par bloc de 32 points.

Bloc B :

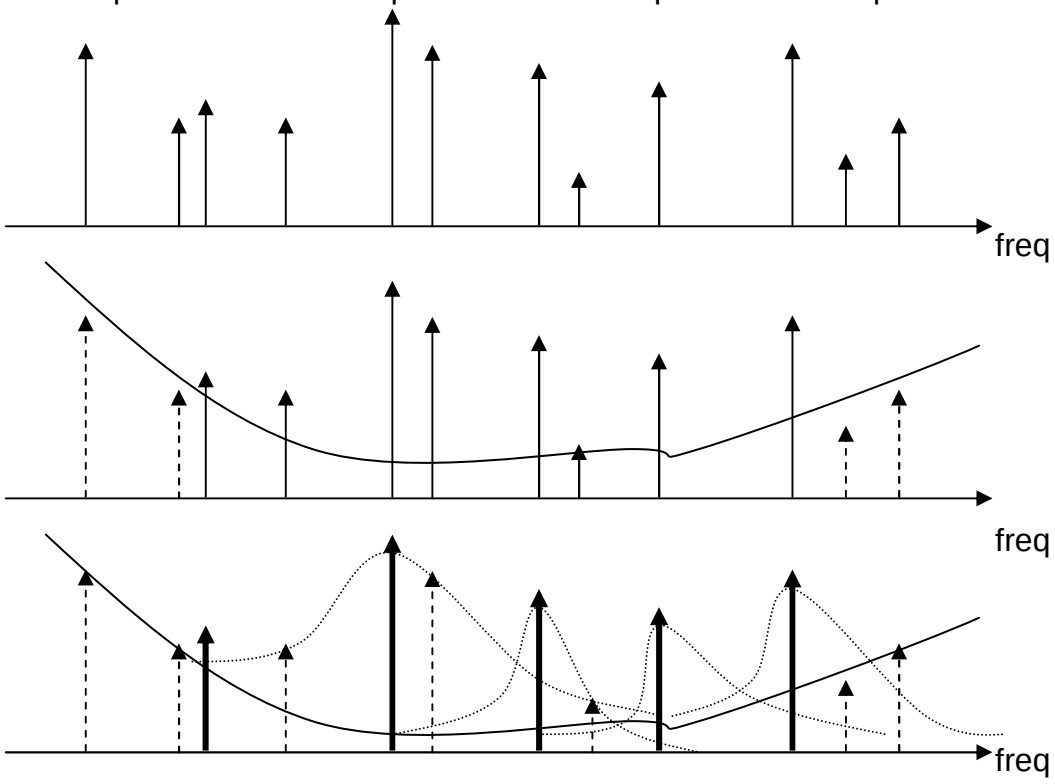
On effectue une TF du signal

On tient compte du seuil d'audition des différentes fréquences pour éventuellement supprimer des composantes non audibles (50 Hz à 30 dB, 10 kHz à 15 dB…)

On calcule l'effet de masquage provoqué par les composantes de forte amplitude. L'effet de masque est l'élévation du seuil d'audition en présence d'un son (masquant) pour des fréquences voisines de celle de ce son.

On supprime les composantes masquées.

Les opérations réalisées par ce bloc sont représentées ci-après :



Bloc C :

On alloue les bits disponibles de façon à avoir dans chaque sous bande un bruit de quantification de niveau inférieur au seuil de masquage.

On minimise le rapport bruit sur masque global

$$\sum_{m=1}^{M} \frac{\sigma_Y^2(m)\, 2^{-2b(m)}}{\sigma^2_{\text{masque}}(m)}$$

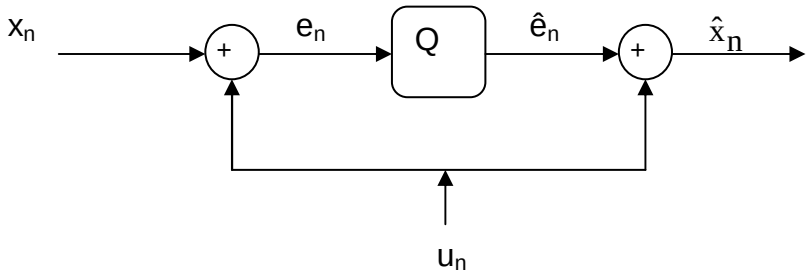
par une procédure itérative du type « algorithme du Greedy ».
Le codage a lieu dans les M = 32 bandes tous les 12 échantillons ce qui donne un nombre de bits disponibles de 12 x 32 x débit / Fe (auxquels on a retranché les bits d'entête , de contrôle de parité, …).

4. Quantification prédictive

La notion de redondance est liée à celle de prédictibilité. Lorsque le signal est redondant il est possible de prédire un échantillon à partir des échantillons passés. L'idée du codage prédictif est alors de ne quantifier et coder que la partie non prédictible du signal.

4.1 Quantification d'une différence

Avant d'utiliser la prédiction nous allons établir un résultat trivial mais important. Supposons que l'on retranche une séquence U_n au processus d'entrée X_n, et que la différence E_n soit quantifiée et que l'on construise $\hat{X}_n$ en rajoutant U_n à $\hat{E}_n$.



On a évidemment : $q_n = e_n - \hat{e}_n$ erreur de quantification

$= \overline{q}_n = x_n - \hat{x}_n$ erreur de reconstruction

On a intérêt à effectuer cette opération si U_n est bien choisi, en particulier si U_n est tel que $E_n = X_n - U_n$ correspond à un processus $E_n = X_n - U_n$ de variance inférieure à celle de X_n (pour un même RSB on pourra diminuer la résolution b).
Un moyen de réaliser ceci est de choisir U_n comme prédiction $\tilde{X}_n$ de X_n.

4.2 Quantification prédictive

Pour une réalisation x_n on considère :

$\tilde{x}_n = -\sum_{i=1}^{P} a_i x_{n-i}$ est la prédiction linéaire de x_n en fonction des P échantillons passés.

On a alors :

$e_n = x_n - \tilde{x}_n = x_n + \sum_{i=1}^{P} a_i x_{n-i}$ l'erreur de prédiction.

On cherche à déterminer les coefficients a_i qui maximisent le rapport signal sur bruit (en linéaire).

$$RSB_{lin} = \frac{E(X_n^2)}{E\left((X_n - \hat{X}_n)^2\right)} = \frac{E(X_n^2)}{E\left((E_n - \hat{E}_n)^2\right)} = \frac{E(X_n^2)}{E(E_n^2)} \frac{E(E_n^2)}{E\left((E_n - \hat{E}_n)^2\right)}$$

$$= \frac{E(X_n^2)}{E(E_n^2)} . RSB_{quant(E_n)}$$

$$= \frac{\sigma_X^2}{\sigma_E^2} RSB_{quant(E_n)}$$

Le second terme dépend du type de quantification de l'erreur de prédiction. Pour une quantification optimale haute résolution b on obtient $\frac{2^{2b}}{C}$.

Comme la puissance du signal σ_X^2 est imposée on peut maximiser le RSB total en minimisant σ_E^2 la puissance moyenne de l'erreur de prédiction.

On aura alors maximisé le gain de prédiction $\boxed{G_P = \frac{\sigma_X^2}{\sigma_E^2}}$.

Nous rappelons ci-après les équations de la prédiction linéaire (la minimisation de la variance de l'erreur de prédiction) et l'algorithme de Levinson pour mettre en œuvre le calcul des coefficients de prédiction.

4.3 Rappels concernant la prédiction linéaire

4.3.1 Equations normales (ou de Yule Walker ou de Wiener Hopf)

On les obtient en minimisant la variance de l'erreur de prédiction

$$\sigma_{E^2} = E(E_n^2) = E\left((X_n + \sum_{i=1}^{P} a_i X_{n-i})^2 \right)$$

En utilisant les notations vectorielles $\underline{a} = \begin{vmatrix} a_1 \\ \vdots \\ a_p \end{vmatrix}$ et $\underline{X}(n) = \begin{vmatrix} X_{n-1} \\ \vdots \\ X_{n-P} \end{vmatrix}$

On a : $\sigma_{E^2} = E\left((X(n))^2\right) + \underline{a}^T E\left(\underline{X}(n)\underline{X}^T(n)\right)\underline{a} + 2\underline{a}^T E\left(\underline{X}(n)X(n)\right)$

Qui s'écrit encore :

$$\sigma_{E^2} = \sigma_{X^2}\left(1 + \underline{a}^T \underline{\underline{\Gamma}}_X(P)\underline{a} + 2\underline{a}^T \rho_X(P)\right)$$

avec $\underline{\underline{\Gamma}}_X(P) = \begin{pmatrix} 1 & \rho_1 & \cdots & \rho_{P-1} \\ \rho_1 & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ \rho_{P-1} & \cdots & \cdots & 1 \end{pmatrix}$, $\rho_{\underline{X}}(P) = \begin{vmatrix} \rho_1 \\ \vdots \\ \rho_P \end{vmatrix}$

et $\rho_i = \dfrac{E(X_n X_{n-i})}{\sigma_{X^2}}$ coefficient d'autocorrélation normalisé

d'où $\dfrac{\partial \sigma_{E^2}}{\partial \underline{a}} = 0$

$\Leftrightarrow$ $\boxed{\underline{\underline{\Gamma}}_X(P)\underline{a} = -\rho_X(P)}$ Cette équation matricielle est l'équation de Wiener Hopf.

Les coefficients a_i optimaux sont obtenus par l'inversion de la matrice d'autocorrélation.

On a alors la variance σ_{E^2} maximale.

$$\sigma_{E^2-\min} = \sigma_{X^2}\left(1 + \underline{a}^T \rho_X(P)\right)$$

$$\boxed{\sigma_{E^2-\min} = \sigma_{X^2}\left(1 + \sum_{i=1}^{P} a_i \rho_i\right)}$$

4.3.2 Algorithme de Levinson (rappels)

C'est un algorithme récursif sur l'ordre du modèle réclamant $O(p^2)$ opérations (au lieu de $O(p^3)$ avec des méthodes classiques).
Si on rassemble les deux équations précédentes le système global de détermination des coefficients de prédiction à l'ordre m et de la variance de l'erreur de prédiction résultante s'écrit :

$$\sigma_X^2 \underline{\underline{\Gamma}}_X (m+1) \begin{vmatrix} 1 \\ \underline{a(m)} \end{vmatrix} = \begin{vmatrix} \sigma_E^2(m) \\ \underline{O} \end{vmatrix} \qquad ①$$

L'algorithme repose sur l'exploitation de la forme de $\underline{\underline{\Gamma}}_X$ qui est une matrice de Toeplitz (les diagonales sont formées d'éléments identiques).

Pour une matrice de cette forme, il est équivalent d'inverser les lignes (multiplication à gauche par $J = \begin{pmatrix} 0 & . & . & 1 \\ & & \iddots & \\ 1 & & & 0 \end{pmatrix}$) et d'inverser les colonnes (multiplication à droite par J).

On a donc si le système ① est vérifié :

$$\sigma_{X_2} \underline{\underline{\Gamma}}_X (m+1) \begin{vmatrix} J\underline{a}(m) \\ 1 \end{vmatrix} = \begin{vmatrix} \underline{0} \\ \sigma_{E_2}(m) \end{vmatrix} \qquad ②$$

Pour passer de l'ordre m à l'ordre m+1 on cherche *<u>a</u>(m+1)* et $\sigma_{E^2}(m+1)$ tels que

$$\sigma_{X^2} \underline{\Gamma}_X (m+2) \begin{vmatrix} 1 \\ \underline{a}(m+1) \end{vmatrix} = \begin{vmatrix} \sigma_{E^2}(m+1) \\ \underline{o} \end{vmatrix}$$

Si on observe que $\left(\begin{array}{c|c} \underline{\underline{\Gamma}}_X (m+1) & \begin{matrix} \rho_{m+1} \\ \vdots \\ \rho_1 \end{matrix} \\ \hline \rho_{m+1} \cdots \rho_1 1 \end{array} \right) = \begin{pmatrix} 1 \rho_1 \cdots \rho_{m+1} \\ \rho_1 \\ \vdots \quad \overline{\left| \underline{\underline{\Gamma}}_X (m+1) \right|} \\ \rho_{m+1} \end{pmatrix}$

On voit qu'il est astucieux d'écrire la solution sous la forme :

$$\boxed{\begin{vmatrix} 1 \\ \underline{a}(m+1) \end{vmatrix} = \begin{vmatrix} 1 \\ \underline{a}(m) \\ 0 \end{vmatrix} + k_{m+1} \begin{vmatrix} 0 \\ J\underline{a}(m) \\ 1 \end{vmatrix}}$$

et de chercher le coefficient k_{m+1} qui convienne.

En effet, si on applique $\underline{\underline{\Gamma}}_X (m+2)$ à $\begin{vmatrix} 1 \\ \underline{a}(m+1) \end{vmatrix}$ ainsi défini

$$\sigma_X^2 \underline{\underline{\Gamma}}_X (m+2)\begin{vmatrix}1\\ \underline{a}(m+1)\end{vmatrix} = \begin{vmatrix}\sigma_{E^2}(m)\\ \underline{0}\\ \underbrace{(\rho_{m+1}+\sum_{i=1}^{m} a_i(m)\rho_{m+1-i})\sigma_X^2}_{\delta(m+1)\sigma_X^2}\end{vmatrix} + k_{m+1}\begin{vmatrix}\delta(m+1)\sigma_X^2\\ \underline{0}\\ \sigma_{E^2}(m)\end{vmatrix}$$

on a alors la condition :

$\sigma_X^2 \delta(m+1) + k_{m+1}\sigma_{E^2}(m) = 0$

$$\Leftrightarrow \qquad \boxed{k_{m+1} = -\frac{\delta(m+1)\sigma_X^2}{\sigma_{E^2}(m)} = -\frac{\left(\rho_{m+1}+\sum_{i=1}^{m} a_i(m)\rho_{m+1-i}\right)\sigma_X^2}{\sigma_{E^2}(m)}}$$

et le résultat :

$\sigma_{E^2}(m+1) = \sigma_{E^2}(m) + k_{m+1}\delta / \sigma_X^2$

$$\boxed{\sigma_{E^2}(m+1) = \sigma_{E^2}(m)(1-k_{m+1}{}^2)}$$

L'algorithme de Levinson est donc le suivant :

$a_1(1) = k_1 = -\rho_1$

$\sigma_E^2(1) = (1-k_1^2)\sigma_X^2$

<u>puis pour m = 2 à P</u>

$$\delta_{m+1} = \rho_{m+1} + \sum_{i=1}^{m} a_i(m)\rho_{m+1-i}$$

$$k_{m+1} = -\delta_{(m+1)}\frac{\sigma_X^2}{\sigma_{E^2}(m)}$$

- pour i = 1 … m
  $a_i^{(m+1)} = a_i^{(m)} + k_{m+1} a_{m+1-i}^{(m)}$
- $a_{m+1}^{(m+1)} = k_{m+1}$
- $\sigma_E^2(m+1) = \sigma^2{}_E(m)(1-k_{m+1}^2)$

4.3.3 Calcul du gain de prédiction : $\frac{\sigma_X^2}{\sigma_E^2(P)} = G_P(P)$

D'après la dernière relation de l'algorithme de Levinson et l'initialisation $\sigma_E^2(0) = \sigma_X^2$

$\sigma_{E_2}(P) = \sigma_X^2 \prod_{m=1}^{P} (1 - k_m^2)$

Le gain de prédiction est donc égal à :

$$G_P(P) = \frac{1}{\prod_{m=1}^{P} (1 - k_m^2)}$$

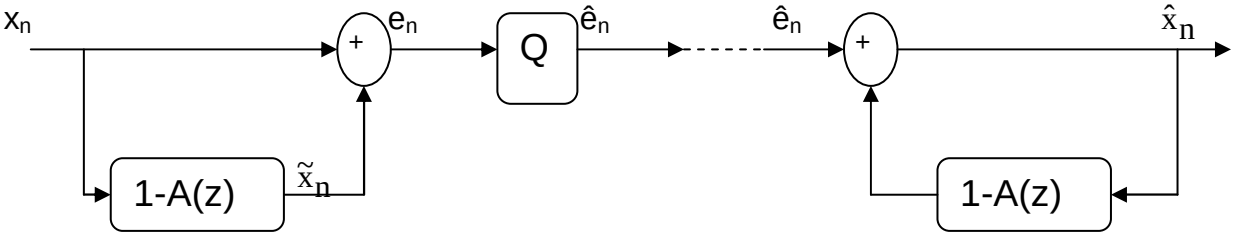
Ce gain est d'autant plus imprtant que les coefficients de réflexion k_m sont de module proche de l'unité.

Mise en œuvre :
on applique le critère des moindres carrés sur l'intervalle des N points connus $x_0 \ldots x_{N-1}$
La forme des équations est alors identique à celle obtenue précédemment en remplaçant

ρ_i par $\hat{\rho}_i = \frac{\sum_{n=i}^{N-1} x_n x_{n-i}}{\sum_{n=0}^{N-1} x_n^2}$

4.4 Quantification scalaire prédictive en boucle ouverte

Le schéma de quantification d'une différence entre le signal x_n et sa prédiction $\tilde{x}_n$ est le suivant :



Si on ne transmet pas $\tilde{x}_n$ on a un cumul de l'erreur de quantification.

En effet si on appelle $q_n = \hat{e}_n - e_n$ cette erreur

$$\hat{X}(z)=\frac{\hat{E}(z)}{A(z)}=\frac{E(z)}{A(z)}+\frac{Q(z)}{A(z)}$$

$$\boxed{\hat{X}(z)=X(z)+\frac{Q(z)}{A(z)}}$$

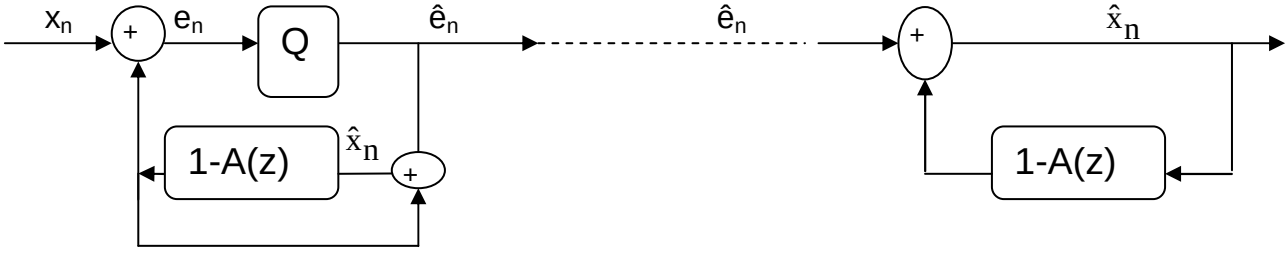
si $r_n = \hat{x}_n - x_n$ est l'erreur de reconstruction

$$R(z)=\frac{Q(z)}{A(z)} \quad \Leftrightarrow \quad \boxed{r_n = q_n - \sum_{i=1}^{P} a_i r_{n-i}}$$

Ceci n'est évidemment pas acceptable, et la solution de transmettre $\tilde{x}_n$ ne l'est pas non plus car elle fait perdre tout le bénéfice du gain de prédiction.
Il faut donc réaliser cette quantification prédictive en boucle fermée c'est-à-dire en incorporant le décodeur dans le codeur.

4.5 Quantification prédictive en boucle fermée

Le nouveau schéma est le suivant :



C'est-à-dire que l'on fait agir le prédicteur 1-A(z) sur le signal reconstruit $\hat{x}_n$
On a bien alors :

$$\hat{E}(z)=E(z)+Q(z)=X(z)-(1-A(z))\hat{X}(z)+Q(z)$$
$$=X(z)-\hat{X}(z)+A(z)\hat{X}(z)+Q(z)$$

$\Leftrightarrow$ $\boxed{\hat{X}(z)=X(z)+Q(z)}$ il n'y a plus de cumul des erreurs.

Ce nouveau schéma pose tout de même quelques problèmes :

- Le e_n n'est pas le résiduel décrit précédemment puisque la prédiction est appliquée sur $\hat{x}_n$ et sera à priori un peu moins performante que lorsqu'elle est faite sur x_n.
- Il faut résoudre le problème du calcul de A(z).

Il y a deux solutions en pratique pour aborder ce second problème :

- Soit on calcule A(z) à partir de x_n (prédiction forward) comme précédemment : On effectue le calcul des $\hat{\rho}_i \quad i=0...P$ et on applique l'algorithme de Levinson.
  - Il faut alors transmettre les coefficients du filtre A(z)
  - le gain de prédiction est un peu inférieur au gain théorique
  - Il y a un retard de N points (~ 20 ms) introduit dans le codage-décodage

- Soit on calcule A(z) à partir de $\hat{x}_n$ (prédiction backward) mais ceci ne peut se faire que de façon adaptative (puisqu'on a besoin de A(z) pour construire $\hat{x}_n$).
  - Il n'est alors pas nécessaire de transmettre A(z)
  - Il n'y a pas de retard introduit
  - Le calcul de A(z) est plus complexe à mettre en œuvre et le gain de prédiction est plus faible que pour la prédiction forward.

4.5.1 Codeur ADPCM (32 kbps)

Ce schéma de quantification prédictive avec prédiction backward est au cœur du codeur MICDA (ADPCM) à 32 kbps (**norme G 721 de l'UIT-T**).

Ce codeur présente quelques particularités supplémentaires :

- Le filtre 1/A(z) est remplacé par un filtre ARMA B(z)/A(z) qui est calculé par un algorithme du gradient simplifié
- Le pas de quantification est adaptatif lui aussi, c'est-à-dire que pour conserver un RSB constant on fait varier le pas de quantification Δ comme l'écart type à court terme du signal.

4.5.2 Codeurs de type RELP (Residual Excited Linear Prediction)

Le principe de ces codeurs repose sur une prédiction forward mais le résiduel e_n n'est pas directement quantifié pour être transmis. On sous-échantillonne l'information e_n avant de la quantifier pour avoir un débit plus faible en transmission. Le décodeur doit bien sûr sur-échantillonner l'excitation transmise avant de la filtrer par 1/A(z).
Ce type de codeur permet des débits de l'ordre de 12 à 16 kbps.
La norme IUT-T du codeur GSM plein débit repose sur une structure de codage de type RELP.

4.5.3 Codeurs de type CELP

Ces codeurs sont eux aussi basés sur la prédiction linéaire (codeur prédictif) et une prédiction généralement forward mais l'information du filtre et l'information du résiduel sont quantifiées vectoriellement avant d'être transmises. Le chapître suivant décrit la technique de **Quantification vectorielle**.

5. Quantification vectorielle

Le principe est de grouper plusieurs échantillons dans un vecteur et de quantifier l'ensemble. Cette technique permet de prendre directement en compte la corrélation contenue dans le signal (contrairement aux techniques de quantification par transformée ou prédictive qui procèdent en deux étapes : décorrélation puis quantification).

Définitions :

- On appelle quantificateur vectoriel de dimension N et de taille L une application :
$$\begin{cases} \Re^N \rightarrow C = \{y_1, y_2, \ldots, y_L\} \subset \Re^N \\ x \rightarrow Q(x) = y_i \end{cases}$$
- L'espace $\Re^N$ est partitionné en L classes
$R_i = \{x / Q(x) = y_i\}$
- *C* est le dictionnaire, les y_i sont des représentants
- Il faut définir une mesure de distorsion liée à l'erreur de quantification *x - Q(x)*

  On choisit habituellement :
$$d(x,Q(x)) = \frac{1}{N}(x-Q(x))^T (x-Q(x)) = \frac{1}{N}\|x-Q(x)\|^2$$
(distance euclidienne)

  ou bien plus généralement une distance pondérée :
$$d(x,Q(x)) = \frac{1}{N}(x-Q(x))^T W (x-Q(x))$$

- On choisit généralement
  $L = 2^{bN}$ b représente la résolution ou nombre de bits par échantillon
  b peut ne pas être entier, il suffit que bN le soit.

5.1 Performances d'un quantificateur vectoriel

Le critère habituellement utilisé est la distorsion moyenne
$$D = E(d(X,Q(X))) = \int_{\Re^N} d(x,Q(x)) f_X(x) dx$$
$$= \sum_{i=1}^{L} \int_{x \in R_i} d(x,y_i) f_X(x) dx$$

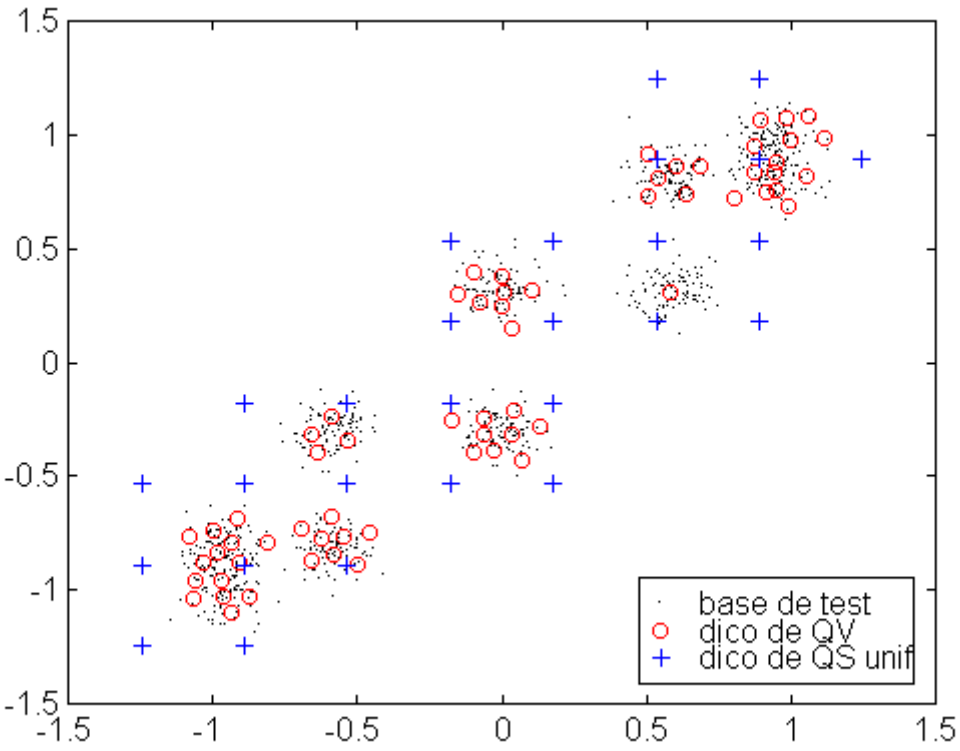
Exemple en dimension 2 :

pour $x \begin{vmatrix} x_{2k} \\ x_{2k+1} \end{vmatrix}$ avec $x_n = A\cos 2\Pi n f_0 + W_n \qquad f_0 = \frac{1}{20}$

et W_n un bruit blanc tel que le RSB soit égal à 20 dB

Etudions la quantification sur 6 bits (3 bits par échantillon)

Dans le cas scalaire : on quantifie chacune des composantes de x sur 3 bits
Les représentants utilisés sont notés par '+'
Dans le cas vectoriel : on quantifie globalement x sur 6 bits , les représentants sont notés par 'o'



Il est évident qu'on obtiendra une distorsion moyenne plus faible dans ce dernier cas.

5.2 Quantificateur vectoriel « optimal »

On cherche à trouver le dictionnaire et les classes associées de façon à minimiser la distorsion moyenne D. Comme dans le cas scalaire on ne peut pas résoudre ce problème globalement mais on conserve les deux conditions nécessaires d'optimalité.

1. Etant donné un dictionnaire C = {y_1, …, y_L} la meilleure partition est celle qui vérifie
   $R_i = \{x / d(x, y_i) \le d(x, y_j) \forall\}$
   C'est la règle du plus proche voisin.
   La partition est appelée partition de Voronoï
   On appelle R_i une cellule de Voronoï (ou région de Voronoï).
2. Etant donnée une partition, les meilleurs représentants sont les centroïdes des cellules de Voronoï
   $$y_i = \frac{\int_{R_i} x f_X(x) dx}{\int_{R_i} f_X(x) dx} = E(X / X \in R_i)$$

L'algorithme de Lloyd – Max généralisé construit un dictionnaire de Q.V de manière itérative en vérifiant tour à tour les deux conditions d'optimalité C_1 et C_2 sur une base d'apprentissage (on l'appelle aussi l'algorithme des K-moyennes (K-means)).

- Dictionnaire initial : $C^{(0)}$ aléatoire
- A l'étape m :dico $C^{(m)}$ $\rightarrow C_1 \rightarrow$ classes $R_i^{(m)}$ Distorsion $D^{(m)}$ $\rightarrow C_2 \rightarrow$ dico $C^{(m+1)}$ $y_i^{(m+1)} = \overline{X}_{R_i^{(m)}}$ (centroïde de $R_i^{(m)}$)

  On arrête lorsque la distorsion n'évolue plus de façon significative :
  $\frac{D(m)-D(m+1)}{D(m)} <$ seuil pré-fixé

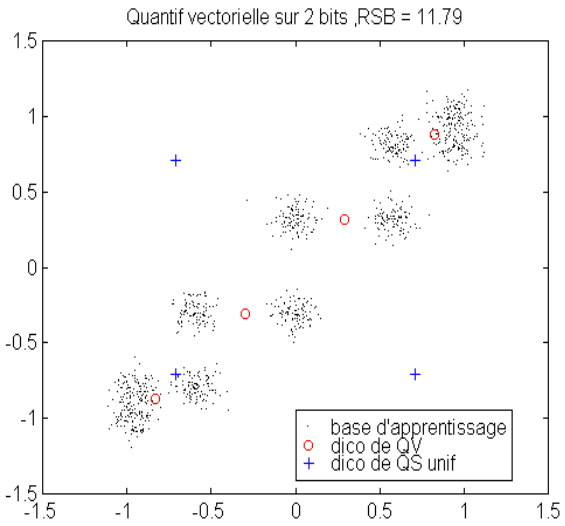
5.3 Algorithme de Linde Buzo Gray (LBG)

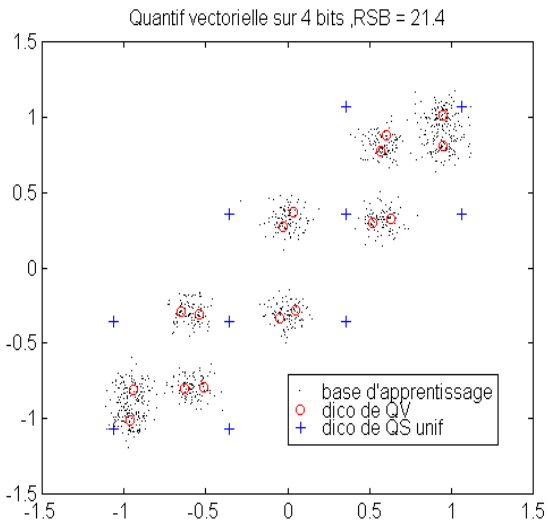
Cet algorithme permet :

- De résoudre le problème du choix du dictionnaire initial
- De construire des dictionnaires de taille 2^k (k = 1 → bN) qui se prêteront à une quantification vectorielle arborescente (par choix successifs entre 2 représentants)
- D 'accélérer la construction du dictionnaire (qui ne se fait qu'une fois pour une base d'apprentissage donnée) mais aussi la quantification (si on utilise la structure arborescente des dictionnaires successifs).

1. Le dictionnaire initial est formé d'un seul vecteur $y_1^{(0)}$ qui est le centroïde de toute la base
2. On éclate ce vecteur en deux $y_1^{(1)}=y_1^{(0)}+\varepsilon$ $y_2^{(1)}=y_1^{(0)}-\varepsilon$ et on calcule les classes $R_1^{((1)}R_2^{(1)}$ associées à ces vecteurs avec la règle du plus proche voisin. On calcule de nouveaux représentants $y_1^{(1)}$ et $y_2^{(1)}$ comme centroïdes de ces classes. On recommence jusqu'à stabilisation de la distorsion (algorithme des 2_moyennes).
3. On éclate les vecteurs stabilisés en 2
   $y_1^{(1)} \pm \varepsilon'$ $y_2^{(1)} \pm \varepsilon''$ $\rightarrow$ $y_1^{(2)} y_2^{(2)} y_3^{(2)} y_4^{(2)}$
   On applique l'algorithme des 4_moyennes sur ces vecteurs.
4. A l'étape k on éclate les 2^{k-1} vecteurs en 2 et on applique l'algorithme des k_moyennes.
5. On s'arrête pour k = bN

Exemple : étape 2 et 4 de l'algorithme LBG pour la sinusoïde bruitée





5.4 Quantification vectorielle de type forme-gain

L'idée est de séparer dans deux dictionnaires distincts ce qui est caractéristique de la forme, le contenu spectral du vecteur et ce qui est caractéristique du gain, l'énergie du vecteur.

Soit : $x \begin{vmatrix} x_1 \\ \vdots \\ x_N \end{vmatrix}$ le vecteur à quantifier

On considère un dictionnaire de formes $y_1 \begin{vmatrix} y_{11} \\ \vdots \\ y_{1N} \end{vmatrix} \cdots y_L \begin{vmatrix} y_{L1} \\ \vdots \\ y_{LN} \end{vmatrix}$

et un dictionnaire de gains $g_1 \ldots g_M$

et $Q(x) = g_i y_j$ doit à priori être recherché de manière exhaustive (en parcourant l'ensemble des valeurs (i, j)). Le principe de la Q.V forme gain est de quantifier x en deux étapes (au prix d'une performance éventuellement moindre). On recherche d'abord parmi toutes les formes y_j la plus proche de celle de x puis on quantifie le gain.

$$d(x, Q(x)) = (x - g_i y_j)^T (x - g_i y_j)$$

Si on dérive par rapport à g_i : $(x - g_i y_j)^T y_j = 0$

$$\Leftrightarrow g_i = \frac{x^T y_j}{\|y_j\|^2}$$

on a alors :

$$\begin{aligned} d_{\min}(x, Q(x)) &= (x - g_i y_j)^T x \\ &= x^T x - g_i y_j^T x \\ &= x^T x - \frac{(y_j^T x)^2}{\|y_j\|^2} \end{aligned}$$

- la forme y_j qui minimise d_{min} est celle qui maximise $\frac{(y_j^T x)^2}{y_j^T y_j}$

c'est-à-dire celle qui maximise $\left| \frac{y_j^T}{\|y_j\|} . x \right| = \|x\| \cos \phi_j |$

(celle qui présente l'angle ϕ_j minimum avec x)

- le gain est alors donné par : $g_i = \frac{x^T y_j}{\|y_j\|^2}$

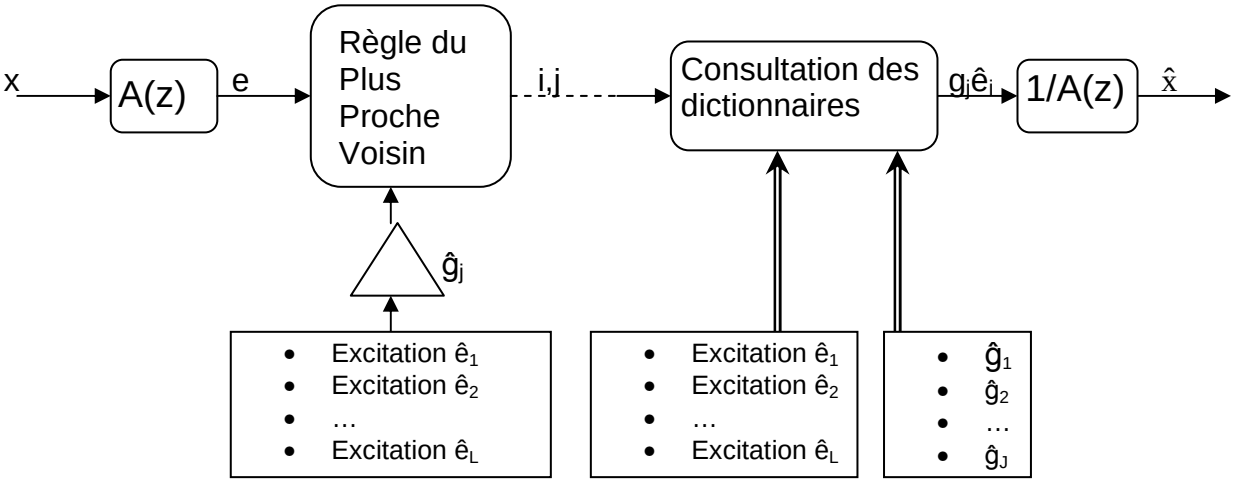
5.5 Principe du codeur CELP (Code Excited Linear Predictive Coder)

Ces codeurs sont utilisés dans la transmission faible et moyen débit de la parole : norme GSM demi débit, norme UMTS…

L'idée est d'utiliser

- un filtre prédictif qui décorrèle (à court terme) les échantillons et fournit une erreur de prédiction (ou résiduel)
- une quantification vectorielle (de type forme-gain) pour coder ce résiduel.

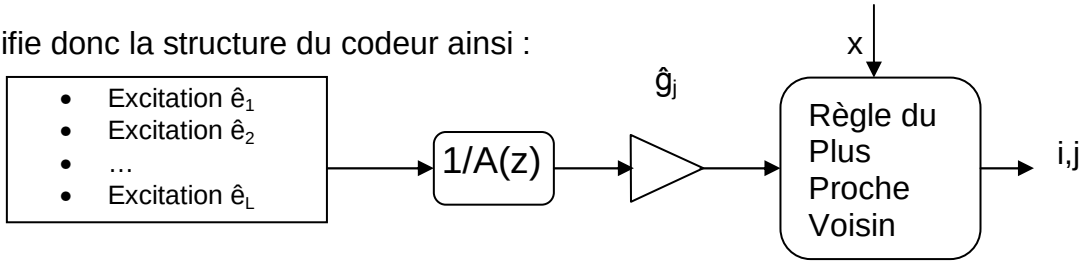
On peut schématiser cette idée comme suit :



En réalité on n'utilise pas ce schéma car :

- c'est un schéma en boucle ouverte
- le plus proche voisin $g_j\hat{e}_i$ de e n'engendre pas forcément le plus proche voisin $\hat{x}$ de x .

On modifie donc la structure du codeur ainsi :



et le décodeur reste inchangé.