# TD 1

# TECHNIQUES DE CODAGE ET DE COMPRESSION

## 1. LANGAGE / CODAGE / VALENCE

### 1.1. Rappels

Toute fraction intelligible d'un message est constituée de ***symboles***. Le ***langage*** est l'ensemble de ces symboles.

Un codage va associer à chaque symbole un ***mot de code***. Chaque mot de code est constitué d'un ensemble de ***signes élémentaires***, encore appelés ***symboles de code***, et est de **longueur $l_i$** correspondant au nombre de signes élémentaires qui le décrivent. La ***valence*** du codage est le nombre de symboles de code. La ***taille*** d'un codage est le nombre de mots de code.

Si tous les mots de code n'ont pas tous la même longueur, le codage est ***de longueur variable***. Le nombre de mots de code associés à un langage dont la longueur est égale à $p$ est alors noté $L_p$.

Un codage est :

- ***intelligible*** si la suite des signes élémentaires correspondant à une succession de symboles possède une unique interprétation ;
- ***instantané*** s'il est possible de le décoder au fur et à mesure de l'arrivée des mots de code ;
- ***préfixe***, si aucun mot de code n'est le début d'un autre mot de code ;
- ***complet*** s'il est intelligible et si tout ajout d'un mot de code de longueur inférieure ou égale à $n$ le rend inintelligible et $L_p = 0$ pour tout p > n.

Une condition nécessaire pour qu'un codage de valence $V$ (dont les mots de code ont une longueur maximale $n$) soit complet et intelligible est donnée par l'égalité de Kraft-McMillan :

$$\sum_{p=1}^{n} \frac{L_p}{V^{p-1}} = V$$

### 1.2. Exercices

Soit le langage représentant les quatre symboles *A, C, G, T*. On considère le codage suivant : *A* est représenté par le mot de code « 0 », *C* par « 01 », *G* par « 001 », *T* par « 101 ».

1.2.1. Quels sont les symboles de code ? Quelle est la valence du codage ?

Chaque symbole de code est un bit. Le langage est donc bivalent (de cardinalité $V = 2$).

1.2.2. Quels sont les valeurs $L_1$, $L_2$, $L_3$, $L_4$ ? Est-ce un codage intelligible, préfixe, vérifie t'il l'égalité de Kraft-McMillan ?

$L_1 = 1$, $L_2 = 1$, $L_3 = 2$, $L_4 = 0$.

Le langage n'est pas intelligible : la suite « 001 » peut être interprétée comme AC ou G.

Il n'est pas non plus préfixe puisque le mot de code associé à *A* est préfixe de celui associé à *C* (et à *G*).

On considère maintenant tous les codes bivalents (de valence 2) et de longueur maximale 3.

1.2.3. Ecrire l'égalité de Kraft-McMillan dans ce cas particulier.

*n* = 3 et *V* = 2, la contrainte est alors : $L_1 + \frac{L_2}{2} + \frac{L_3}{4} = 2$.

1.2.4. Résoudre cette équation (donner toutes les valeurs possibles de $L_1$, $L_2$ et $L_3$ vérifiant cette égalité).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $L_1$ | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $L_2$ | 0 | 2 | 1 | 0 | 4 | 3 | 2 | 1 | 0 |
| $L_3$ | 0 | 0 | 2 | 4 | 0 | 2 | 4 | 6 | 8 |

1.2.5. Pour chaque triplet de valeurs, donner un code bivalent asssocié.

Pour les différentes valeurs de $L_1$, $L_2$ et $L_3$ vérifiant la contrainte précédente, on obtient le tableau suivant :

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $L_1$ | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $L_2$ | 0 | 2 | 1 | 0 | 4 | 3 | 2 | 1 | 0 |
| $L_3$ | 0 | 0 | 2 | 4 | 0 | 2 | 4 | 6 | 8 |
| Mots de code | 0<br>1 | 0<br>10<br>11 | 0<br>10<br>110<br>111 | 0<br>100<br>101<br>110<br>111 | 00<br>01<br>10<br>11 | 00<br>01<br>10<br>110<br>111 | 00<br>01<br>100<br>101<br>110<br>111 | 00<br>010<br>011<br>100<br>101<br>110<br>111 | 000<br>001<br>010<br>011<br>100<br>101<br>110<br>111 |
| Taille du codage | 2 | 3 | 4 | 5 | 4 | 5 | 6 | 7 | 8 |

Soit le codage représentant les quatre symboles *A*, *C*, *G*, *T* où *A* est représenté par le mot de code « 0 », *C* par « 100 », *G* par « 101 », *T* par « 111 ».

1.2.6. Quels sont les valeurs $L_1$, $L_2$, $L_3$, $L_4$ ? Est-ce un codage intelligible ? Est-ce un codage complet ?

$L_1 = 1$, $L_2 = 0$, $L_3 = 3$, $L_4 = 0$. Le codage est préfixe donc est intelligible.

Ce codage n'est pas complet, car l'ensemble des mots de code est un sous-ensemble du codage complet de taille 5 : {0, 100, 101, 110, 111}

Les codages {0, 10, 110, 111} et {00, 01, 10, 11} sont complets donc mieux adaptés.

## 2. CODAGES COMPRESSIFS : SHANNON, FANO-SHANNON ET HUFFMAN

Dans toute la suite, sauf mention explicite du contraire, on considère des codes binaires.

## 2.1. Rappels

A chaque symbole $s_i$ est associée une **probabilité** $p_i$ représentant la fréquence d'occurrence du symbole. Ces probabilités sont estimées, soit *a priori* sur des *corpus* (échantillons) représentatifs de l'application considérée, soit *a posteriori* sur le corpus réel de l'application. La première solution ne fournit qu'une estimation, alors que la seconde nécessite un pré-traitement du corpus.

On définit l'***entropie*** d'une source de la façon suivante :

$$H(S) = -\sum_i p_i \log_2(p_i)$$

La base du logarithme fixe l'unité de l'entropie : dans le cas de la base 2, l'entropie s'exprime en ***shannon*** (sh). On peut démontrer que l'entropie est maximum dans le cas de l'équiprobabilité des symboles.

L'efficacité d'un code est estimée par ses caractéristiques :

- **Longueur moyenne** : $l_{moy} = \sum_i l_i \ p_i$. On peut montrer que $\frac{H(S)}{\log_2(V)} \leq l_{moy}$, la limite théorique est donc : $l_{min} = \frac{H(S)}{\log_2(V)}$, ce qui se simplifie si le code est bivalent.
- **Rendement** $R = \frac{l_{min}}{l_{moy}} = \frac{H(S)}{l_{moy}\log_2(V)}$
- **Redondance** : $\rho = 1 - R$

Tous ces paramètres sont relatifs à l'entropie de la source.

## 2.2. Entropie d'une source

Soit une source (S) à 11 symboles ($s_1$ à $s_{11}$) définie par les probabilités suivantes :

| S | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ | $s_{10}$ | $s_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 0.22 | 0.15 | 0.12 | 0.11 | 0.10 | 0.08 | 0.07 | 0.06 | 0.04 | 0.03 | 0.02 |

2.2.1. Calculer l'entropie de la source.

| S | $p_i$ | $-\log_2(p_i)$ | $-p_i \log_2(p_i)$ |
|---|---|---|---|
| $s_1$ | 0.22 | 2.184 | 0.481 |
| $s_2$ | 0.15 | 2.737 | 0.411 |
| $s_3$ | 0.12 | 3.059 | 0.367 |
| $s_4$ | 0.11 | 3.184 | 0.350 |
| $s_5$ | 0.10 | 3.322 | 0.332 |
| $s_6$ | 0.08 | 3.644 | 0.292 |
| $s_7$ | 0.07 | 3.837 | 0.269 |
| $s_8$ | 0.06 | 4.059 | 0.244 |
| $s_9$ | 0.04 | 4.644 | 0.186 |
| $s_{10}$ | 0.03 | 5.059 | 0.152 |
| $s_{11}$ | 0.02 | 5.644 | 0.113 |
| | | | ***H(S)* = 3.197 sh** |

2.2.2. Que vaudrait l'entropie si tous les symboles étaient équiprobables ?

Dans le cas de l'équiprobabilité :

$$p_i = \frac{1}{11}$$

et $\boldsymbol{H(S) = -} \log_2(\frac{1}{11}) = 3.459$

On constate bien que l'entropie est supérieure dans le cas de l'équiprobabilité des symboles.

### 2.3. Codage de Shannon

Les symboles $s_i$ sont codés par un nombre $l_i$ d'éléments unitaires tel que

$$-\frac{\log_2(p_i)}{\log_2(V)} \leq l_i < -\frac{\log_2(p_i)}{\log_2(V)} + 1$$

2.3.1. Donner un code de Shannon binaire pour la source ci-dessus. Une méthode simple consiste à calculer la longueur de chaque mot de code puis à construire un arbre binaire respectant ces longueurs

| S | $p_i$ | | $l_i$ | $l_i\, p_i$ | **Code proposé** |
|---|---|---|---|---|---|
| $s_1$ | 0.22 | $2.184 \le l_1 < 3.184$ | 3 | 0.66 | **000** |
| $s_2$ | 0.15 | $2.737 \le l_2 < 3.737$ | 3 | 0.45 | **001** |
| $s_3$ | 0.12 | $3.059 \le l_3 < 4.059$ | 4 | 0.48 | **0100** |
| $s_4$ | 0.11 | $3.184 \le l_4 < 4.184$ | 4 | 0.44 | **0101** |
| $s_5$ | 0.10 | $3.322 \le l_5 < 4.322$ | 4 | 0.40 | **0110** |
| $s_6$ | 0.08 | $3.644 \le l_6 < 4.644$ | 4 | 0.32 | **0111** |
| $s_7$ | 0.07 | $3.837 \le l_7 < 4.837$ | 4 | 0.28 | **1000** |
| $s_8$ | 0.06 | $4.059 \le l_8 < 5.059$ | 5 | 0.30 | **10010** |
| $s_9$ | 0.04 | $4.644 \le l_9 < 5.644$ | 5 | 0.20 | **10011** |
| $s_{10}$ | 0.03 | $5.059 \le l_{10} < 6.059$ | 6 | 0.18 | **101000** |
| $s_{11}$ | 0.02 | $5.644 \le l_{11} < 6.644$ | 6 | 0.12 | **101001** |
| | | | | $\boldsymbol{l_{moy} = 3.83}$ | |

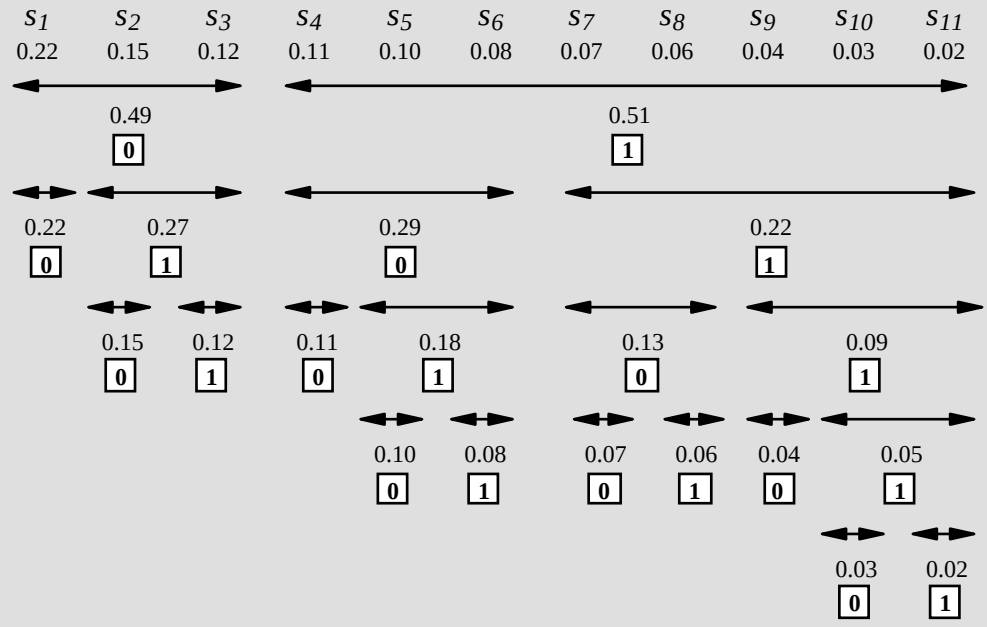
2.3.2. Calculer les caractéristiques de ce code : longueur moyenne, rendement et redondance.

| $\boldsymbol{l_{moy} = 3.83}$ | $\boldsymbol{R = 0.835}$ | $\boldsymbol{\rho = 16.5\%}$ |
|---|---|---|

## 2.4. Codage de Fano-Shannon

Il s'agit de construire un arbre en équilibrant à chaque fois au maximum les sous-arbres droit et gauche.

2.4.1. Donner un code binaire pour la source ci-dessus en appliquant la méthode de Fano-Shannon.




Le codage obtenu est le suivant :

| S | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| code | 00 | 010 | 011 | 100 | 1010 | 1011 | 1100 | 1101 | 1110 | 11110 | 11111 |

2.4.2. Calculer les caractéristiques de ce code.

| S | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ | $s_{10}$ | $s_{11}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 0.22 | 0.15 | 0.12 | 0.11 | 0.10 | 0.08 | 0.07 | 0.06 | 0.04 | 0.03 | 0.02 | |
| $l_i$ | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | |
| $l_i\, p_i$ | 0.44 | 0.45 | 0.36 | 0.33 | 0.40 | 0.32 | 0.28 | 0.24 | 0.16 | 0.15 | 0.10 | **$l_{moy}$ = 3.23** |

| **$l_{moy}$ = 3.23** | **$R$ = 0.990** | **$\rho$ = 1%** |
|---|---|---|

## 2.5. Codage de Huffman

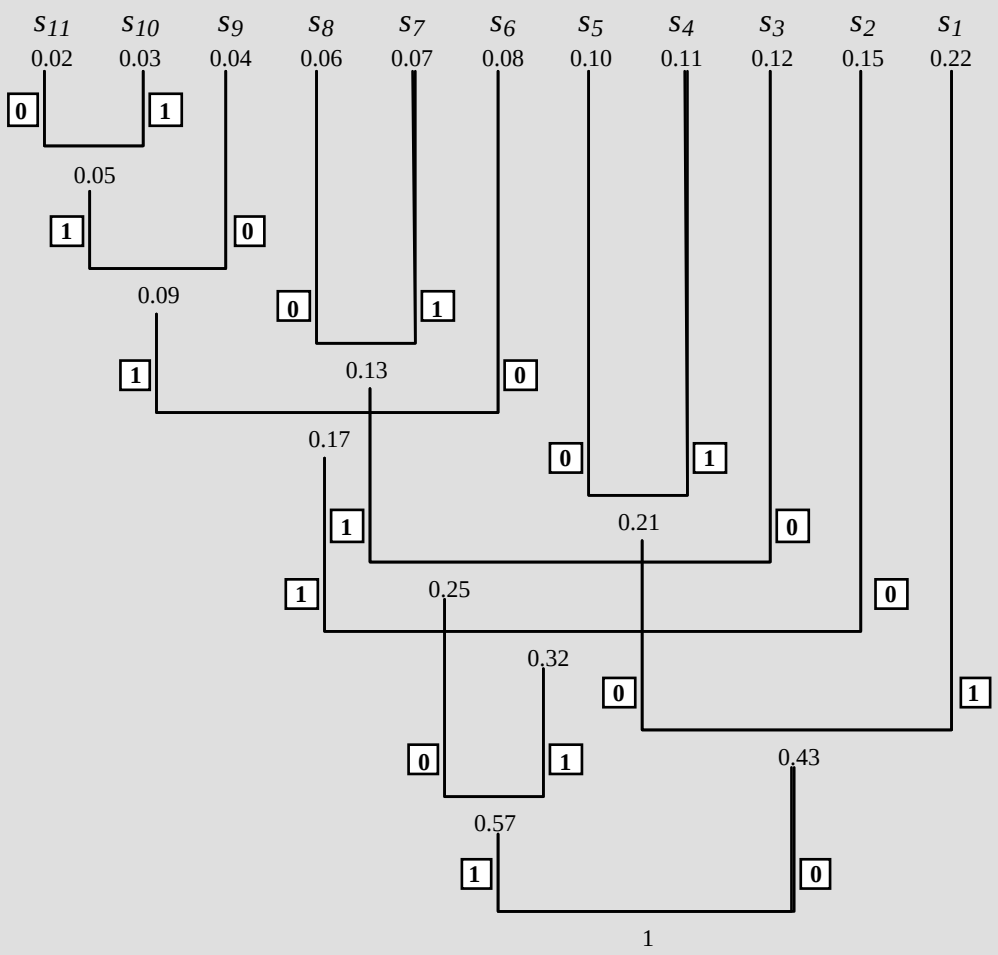
L'algorithme consiste à construire un arbre *V*-aire à partir des feuilles (symboles d'origine) vers la racine :

1. Ordonner les symboles suivant les probabilités croissantes.
2. Rassembler***** les *V* symboles les moins probables en un super-symbole dont la probabilité est la somme des probabilités des symboles intervenant. Adjoindre à chacun des *V* symboles un élément différent de l'alphabet.
3. Si la nouvelle source comporte plus d'un symbole, aller en 2 sinon fin de l'algorithme.

Le code de Huffman des symboles de source s'obtient par un parcours de la racine vers les feuilles.

***Attention :** Pour $V > 2$, si $r = (n-1) \bmod (V-1) \neq 0$ alors au premier passage à l'étape 2, rassembler r+1 symboles.

2.5.1. Donner un code binaire pour la source ci-dessus en appliquant la méthode de Huffman. Par convention, on choisira de coder les symboles ou super-symboles de plus faible probabilité par « 0 ».




Le codage obtenu est le suivant :

| S | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 | s9 | s10 | s11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| code | 01 | 110 | 100 | 001 | 000 | 1110 | 1011 | 1010 | 11110 | 11111 | 11110 |

2.5.2. Calculer les caractéristiques de ce code.

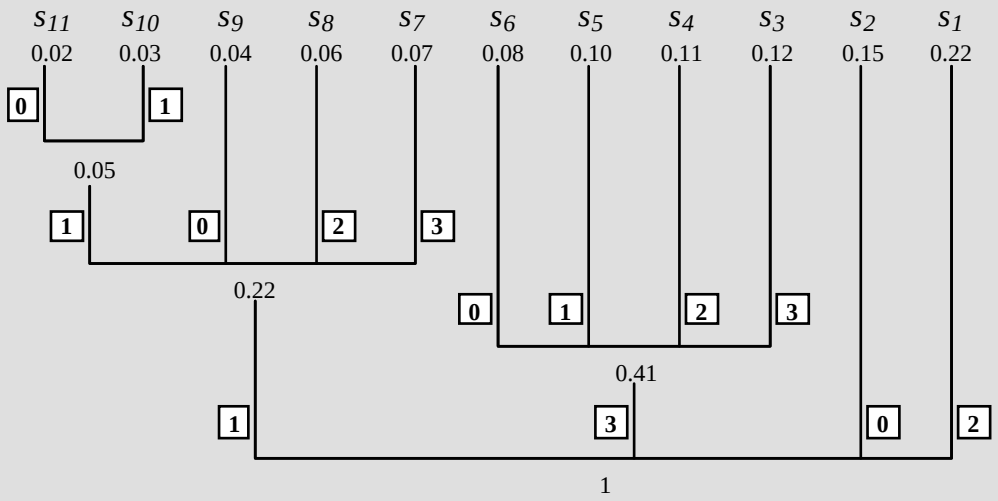
| S | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ | $s_{10}$ | $s_{11}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 0.22 | 0.15 | 0.12 | 0.11 | 0.10 | 0.08 | 0.07 | 0.06 | 0.04 | 0.03 | 0.02 | |
| $l_i$ | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 6 | 6 | |
| $l_i\, p_i$ | 0.44 | 0.45 | 0.36 | 0.33 | 0.30 | 0.32 | 0.28 | 0.24 | 0.20 | 0.18 | 0.12 | $\boldsymbol{l_{moy} = 3.22}$ |

| $\boldsymbol{l_{moy} = 3.22}$ | $\boldsymbol{R = 0.993}$ | $\boldsymbol{\rho = 0.7\%}$ |
|---|---|---|

2.5.3. Soit « 001111001011011 » la séquence interprétée par un décodeur en bande de base, à quel message correspond cette séquence ?

La séquence « 001111001011011 » correspond au message $s_4\ s_6\ s_1\ s_1\ s_7$.

2.5.4. Un code quaternaire trouve son application dans le cadre d'une transmission en bande de base d'un codage de valence 4. Donner un code quaternaire en appliquant la méthode de Huffman en faisant attention à la note plus haut. Quel code préfèrera-t-on (binaire ou quaternaire) ?




Le codage obtenu est le suivant :

| S | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ | $s_{10}$ | $s_{11}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| code | 2 | 0 | 33 | 32 | 31 | 30 | 13 | 12 | 10 | 111 | 110 |

| S | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ | $s_{10}$ | $s_{11}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p_i$ | 0.22 | 0.15 | 0.12 | 0.11 | 0.10 | 0.08 | 0.07 | 0.06 | 0.04 | 0.03 | 0.02 | |
| $l_i$ | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | |
| $l_i\, p_i$ | 0.22 | 0.15 | 0.24 | 0.22 | 0.20 | 0.16 | 0.14 | 0.12 | 0.08 | 0.09 | 0.06 | $l_{moy}$ **= 1.68** |

$$R = \frac{H(S)}{l_{moy}\log_2(V)} \Rightarrow R = \frac{3.197}{2*1.68} = 0.952$$

| $l_{moy}$ **= 1.68** | ***R* = 0.952** | ρ = **4.8%** |
|---|---|---|

| **Code binaire** | ***R* = 0.993** | ρ = **0.7%** |
|---|---|---|
| **Code quaternaire** | ***R* = 0.952** | ρ = **4.8%** |

Les caractéristiques des codes montrent que le codage binaire est mieux adapté à la source que le codage quaternaire.

## 2.6. Comparaison de l'efficacité des codes

2.6.1. Conclure sur l'efficacité des codes binaires. L'un de ces codes est-il absolument optimal ?

**Résumé :**

| | **Shannon** | **Fano-Shannon** | **Huffman** |
|---|---|---|---|
| $H(S) = l_{min}$ **= 3.197** | $l_{moy}$ **= 3.83eb** | $l_{moy}$ **= 3.23eb** | $l_{moy}$ **= 3.22eb** |

| | | | |
|---|---|---|---|
| | ***R* = 0.835** | ***R* = 0.990** | ***R* = 0.993** |
| | **ρ = 16.5%** | **ρ = 1%** | **ρ = 0.7%** |

On note que les trois codes binaires sont **efficaces** ($l_{moy} < H(S)+1$), et de plus en plus efficace suivant leur ordre de présentation. **Aucun** de ces codes n'est **absolument optimal** puisque leur longueur moyenne ($l_{moy}$) est supérieure à la longueur minimum théorique ($l_{min}$)

# 3. CODAGE PAR BLOC DE SYMBOLES

## 3.1. Rappels

L'objectif d'un codage par bloc de symboles est d'améliorer l'efficacité d'un codage par symbole (non optimal) pour se rapprocher de la borne inférieure théorique.

## 3.2. Exercices

On considère 2 sources $S_A$ et $S_B$, chacune constituée de 3 symboles et définies par leurs probabilités :

| $S_A$ | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|
| $P(S_A = s_i)$ | 0.38 | 0.34 | 0.28 |

| $S_B$ | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|
| $P(S_B = s_i)$ | 0.5 | 0.25 | 0.25 |

3.2.1. Calculer l'entropie des deux sources.

| $S_A$ | $p_i$ | $-\log_2(p_i)$ | $-p_i \log_2(p_i)$ |
|---|---|---|---|
| $s_1$ | 0.38 | 1.396 | 0.530 |
| $s_2$ | 0.34 | 1.556 | 0.529 |
| $s_3$ | 0.28 | 1.837 | 0.514 |
| | | | **$H(S_A)$ = 1.573sh** |
| **$S_B$** | **$p_i$** | **$-\log_2(p_i)$** | **$-p_i \log_2(p_i)$** |
| $s_1$ | 0.5 | 1 | 0.5 |
| $s_2$ | 0.25 | 2 | 0.5 |
| $s_3$ | 0.25 | 2 | 0.5 |
| | | | **$H(S_B)$ = 1.5 sh** |

3.2.2. Quel est, parmi les codages de Shannon, Fanno-Shannon et Huffman, le codage le plus efficace pour chacune des deux sources ? Comparer leur efficacité.

Pour la source $S_A$ Les méthodes de Fanno-Shannon et Huffman sont équivalentes. Un code possible est $s_1$ (« 0 »), $s_2$ (« 10 »), $s_3$ (« 11 »). La redondance du code est de 7.3%.

Pour la source $S_B$ Les méthodes de Shannon, Fanno-Shannon et Huffman sont équivalentes. Un code possible est $s_1$ (« 0 »), $s_2$ (« 10 »), $s_3$ (« 11 »). La redondance du code est nulle.

Conclusion : Les méthodes de codage sont moins efficaces quand les probabilités $p_i$ des symboles à coder ne sont pas des puissances négatives de la valence $V$ ($V = 2$ dans l'exemple).

3.2.3. Montrer que le codage de Huffman est optimal si les probabilités $p_i$ des symboles à coder sont des puissances négatives de la valence $V$ du codage.

Les probabilités sont donc $p_1=2^{-1}$, $p_2=2^{-2}$, $p_3=2^{-3}$, …, $p_n=2^{-n}$, $p_{n+1}=2^{-n}$.

Avec la méthode de Huffman, $s_1$ sera codé par 0, $s_2$ par 10, etc. La longueur moyenne est donc égale à l'entropie.

On se propose de coder les symboles 2 par 2. Cela revient à définir une nouvelle source qui émet un bi-symbole. On applique cette transformation à la source $S_A$ en supposant que la probabilité d'un bi-symbole est le produit de la probabilité des symboles le composant.

3.2.4. Donner pour cette nouvelle source $S_A^2$, le codage le plus efficace. Conclure sur l'efficacité du codage par bloc ?

| $S_A^2$ | $p_i$ | $-\log_2(p_i)$ | $-p_i \log_2(p_i)$ |
|---|---|---|---|
| $s_1s_1$ | 0.144 | 2.796 | 0.403 |
| $s_1s_2$ | 0.129 | 2.955 | 0.381 |
| $s_2s_1$ | 0.129 | 2.955 | 0.381 |
| $s_2s_2$ | 0.116 | 3.108 | 0.361 |
| $s_1s_3$ | 0.106 | 3.238 | 0.343 |
| $s_3s_1$ | 0.106 | 3.238 | 0.343 |
| $s_2s_3$ | 0.095 | 3.396 | 0.323 |
| $s_3s_2$ | 0.095 | 3.396 | 0.323 |
| $s_3s_3$ | 0.080 | 3.644 | 0.292 |
| | | | $H(S_A^2) = 3.15$sh |

Pour la source $S_A^2$ Les méthodes de Fanno-Shannon et Huffman sont équivalentes. Un code possible est $s_1s_1$ (« 001 »), $s_1s_2$ (« 010 »), $s_2s_1$ (« 011 »), $s_2s_2$ (« 100 »), $s_1s_3$ (« 101 »), $s_3s_1$ (« 110 »), $s_2s_3$ (« 111 »), $s_3s_2$ (« 0000 »), $s_3s_3$ (« 0001 »). La redondance du code est maintenant de 0.8%. Le codage par bloc est plus efficace.

## 4. CODAGES AVEC PREDICTION POUR LES IMAGES

### 4.1. Rappels

La procédure de modulation par impulsions codées différentielles (MICD) ou Differential Pulse Coding Modulation (DPCM), consiste à calculer d'abord la différence entre le signal d'entrée x et une valeur de prédiction p.

En pratique, p est une combinaison linéaire de x et de ses voisins. Différentes combinaisons existent pour déterminer la valeur de prédiction p. Si $x_{i,j}$ est le pixel considéré alors p peut être défini par $x_{i-}$

$_{1,j}$ ou $x_{i,j-1}$ (valeur du pixel précédent sur la même ligne ou sur la même colonne) ou par une les relations suivantes : $x_{i-1,j-1}$, $x_{i,j-1}+x_{i-1,j}-x_{i-1,j-1}$, $x_{i,j-1}+(x_{i-1,j}-x_{i-1,j-1})/2$, $x_{i-1,j}+(x_{i,j-1}-x_{i-1,j-1})/2$, $(x_{i,j-1}+x_{i-1,j})/2$

Cette valeur de prédiction est connue du décodeur. L'erreur « x–p » est ensuite quantifiée à l'aide d'une matrice de quantification et l'on obtient $e_q$. On code alors en mots binaires ces valeurs par indexation. On reconstruit simplement la valeur codée en ajoutant $e_q$ à la valeur de prédiction.

Lors de la décomposition, la connaissance des pixels reconstruits permet de calculer les valeurs de prédiction d'indices supérieurs, et en leur ajoutant les erreurs de prédiction quantifiées, on reconstruit les pixels de l'image.

Exemple de table de quantification (optimisée par le CCETT)

| Erreur de prédiction | Valeur quantifiée de e : $e_q$ | Erreur de prédiction | Valeur quantifiée de e : $e_q$ |
|---|---|---|---|
| $-255 \le e \le -70$ | - 80 | $9 \le e \le 18$ | 12 |
| $-69 \le e \le -50$ | - 58 | $19 \le e \le 32$ | 25 |
| $-49 \le e \le -33$ | - 40 | $33 \le e \le 47$ | 39 |
| $-32 \le e \le -19$ | - 25 | $48 \le e \le 64$ | 55 |
| $-18 \le e \le -9$ | - 12 | $65 \le e \le 83$ | 73 |
| $-8 \le e \le -3$ | - 4 | $84 \le e \le 104$ | 93 |
| $-2 \le e \le 2$ | 0 | $105 \le e \le 127$ | 115 |
| $3 \le e \le 8$ | 4 | $128 \le e \le 255$ | 140 |

La forme la plus simple du codage prédictif est appelé « linéaire Modulation » ou Delta Modulation. Le prédicteur est une fonction dépendant simplement de la valeur de la donnée précédente, et on utilise un quantificateur sur 1 bit ce qui permet une représentation sur 1 bit du signal.

On peut également utiliser une DPCM bidimensionnelle, comme le fait JPEG en mode sans perte. L'erreur de prédiction n'est pas quantifiée.

### 4.2. Exercices

Si l'on considère que la valeur de prédiction pour les indices i,j est obtenue par $p = (x_{i,j-1} + x_{i-1,j}) / 2$ pour i et j >1. Si i (ou j) = 1 (1ère ligne ou colonne respectivement) $p = x_{i,j-1}$ ( resp. $x_{i-1,j}$), et $x_{1,1}$ est transmis tel quel.

On utilise la table de quantification précédente, la transmission des indexes est codée sur 4 bits (au lieu de 8 pour les données).

Matrice originale

| | | | |
|---|---|---|---|
| 100 | 102 | 106 | 92 |
| 98 | 100 | 104 | 100 |
| 70 | 80 | 92 | 98 |
| 72 | 76 | 84 | 90 |

4.2.1. Calculer la matrice de prédiction avec les règles énoncées plus haut.

Matrice de prédiction

| | | | |
|---|---|---|---|
| | 100 | 102 | 106 |
| 100 | 100 | 103 | 98 |
| 98 | 85 | 92 | 96 |
| 70 | 76 | 84 | 91 |

4.2.2. Calculer la matrice d'erreurs de prédiction.

Matrice d'erreurs de prédiction

| | | | |
|---|---|---|---|
| | 2 | 4 | -14 |
| -2 | 0 | 1 | 2 |
| -28 | -5 | 0 | 2 |
| 2 | 0 | 0 | -1 |

4.2.3. Quantifier la matrice d'erreurs de prédiction à l'aide la table de quantification précédente. La transmission des indexes est codée sur 4 bits au lieu de 8 pour les données (il y a seulement 16 valeurs possible d'erreurs quantifiées).

Matrice des erreurs quantifiées

| | | | |
|---|---|---|---|
| | 0 | 4 | -12 |
| 0 | 0 | 0 | 0 |
| -25 | -4 | 0 | 0 |
| 0 | 0 | 0 | 0 |

4.2.4. Reconstruire l'image (en arrondissant à l'entier supérieur). Quelle est l'erreur moyenne ?

Image reconstruite

| | | | |
|---|---|---|---|
| 100 | 100 | 104 | 92 |
| 100 | 100 | 102 | 97 |
| 75 | 84 | 93 | 95 |
| 75 | 80 | 87 | 91 |

L'erreur moyenne est 35/16 = 2,1875.

# 5. CODAGE EN GROUPE

## 5.1. Rappels

Le block truncation coding ou codage en groupe est une méthode très simple à mettre en œuvre qui opère sur des blocs d'images. On calcule pour chaque bloc la moyenne X et l'écart type $\sigma$. Ces deux valeurs sont transmises ainsi qu'une matrice de signes, qui indique pour chaque point du bloc s'il se trouve au dessus ou en dessous de la moyenne. La valeur reconstruite sera $(X + \sigma)$ ou $(X - \sigma)$ selon le cas.

Cette technique peut être améliorée en schématisant un quantificateur sur un bit. On calcule donc pour chaque bloc n × n = m, la moyenne et l'écart type.

$$\overline{X} = \frac{1}{m}\sum_{i,j} x_{i,j} \qquad \overline{X^2} = \frac{1}{m}\sum_{i,j} x_{i,j}^2 \qquad \sigma^2 = \overline{X^2} - \overline{X}^2$$

Pour une valeur de seuil particulière il attribue la valeur a (resp. b) pour le bloc reconstruit si la valeur correspondante du bloc d'origine est supérieure au seuil (resp. inférieure). Dans le cas particulier où le seuil est la moyenne, les valeurs de a et de b sont calculées en résolvant les équations suivantes :

$$m\overline{X} = (m-q)a + qb \qquad m\overline{X^2} = (m-q)a^2 + qb^2$$

Où q est le nombre de pixels du bloc supérieur au seuil, ce qui donne finalement :

$$a = \overline{X} - \sigma\sqrt{\frac{q}{(m-q)}} \qquad b = \overline{X} + \sigma\sqrt{\frac{(m-q)}{q}}$$

Alors chaque bloc est défini par les valeurs de la moyenne, de l'écart type et d'un plan de nxn bits constitué de 0 et de 1, indiquant si les pixels sont au dessus ou au dessous du seuil.

Cette méthode est utilisé pour la compression d'images en niveaux de gris et réduit le nombre de niveaux.

### 5.2. Exercices

Soit le bloc 4x4 suivant :

$$x_{i,j} = \begin{bmatrix} 121 & 114 & 56 & 47 \\ 37 & 200 & 247 & 255 \\ 16 & 0 & 12 & 169 \\ 43 & 5 & 7 & 251 \end{bmatrix}$$

5.2.1. Calculer la moyenne, l'écart type et q.

$\overline{X} = \frac{1}{m}\sum_{i,j} x_{i,j} = 98{,}75$

$\sigma = 92{,}95$

$q = 7$

5.2.2. En déduire a et b (en arrondissant au plus proche), puis le plan de bits 4x4 et le bloc reconstruit.

$a=16{,}7 \approx 17$ et $b=204{,}2 \approx 204$

le plan de bits est :

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

le bloc est reconstruit en :

$$\begin{bmatrix} 204 & 204 & 17 & 17 \\ 17 & 204 & 204 & 204 \\ 17 & 17 & 17 & 204 \\ 17 & 17 & 17 & 204 \end{bmatrix}$$

5.2.3. Que préserve cette méthode ?

La variance et la moyenne sont préservées aux arrondis près.

# 6. AUTRES CODAGES

## 6.1. Le codage à base de dictionnaires (Lempel-Ziv-Welch)

Initialisation : un dictionnaire est créé, typiquement avec tous les caractères du code ASCII.

Les caractères sont lus un par un (c) et stockés dans un buffer (B) :

- Si B+c est dans le dictionnaire alors B=B+c
- Sinon B est envoyé, B+c est ajouté dans le dictionnaire, B=c

A la fin le buffer est renvoyé s'il n'est pas vide (il correspond forcément à un mot du dictionnaire).

## 6.2. Exercices

6.2.1. Choisir un mot quelconque contenant des répétitions de groupes de lettres et le coder avec l'algorithme LZW (par exemple le mot « DAD DADA DADDY DO »).

| Caractère | Buffer | Emission | Index dans le dictionnaire | Nouvelle chaine du dictionnaire |
|---|---|---|---|---|
| D | D | / | / | / |
| A | DA | D | 256 | DA |
| D | AD | A | 257 | AD |
| _ | D_ | D | 258 | D_ |
| D | _D | _ | 259 | _D |
| A | DA | / | / | / |
| D | DAD | 256 | 260 | DAD |
| A | DA | / | / | / |
| _ | DA_ | 256 | 261 | DA_ |
| D | _D | / | / | / |
| A | _DA | 259 | 262 | _DA |
| D | AD | / | / | / |
| D | ADD | 257 | 263 | ADD |

| | | | | |
|---|---|---|---|---|
| Y | DY | D | 264 | DY |
| _ | Y_ | Y | 265 | Y_ |
| D | _D | / | / | / |
| O | _DO | 259 | 266 | _DO |
| / | O | O | / | / |

6.2.2. Proposer un algorithme de décodage et l'appliquer.

La chaine lue est D ;A ;D ;_ ;256 ;256 ;259 ;257 ;D ;Y ;259 ;0

Sortie D, puis A (ajout DA dans le dico avec 256), puis D (ajout AD dans le dico avec 257), puis _ (ajout D_ dans le dico avec 258), puis DA (ajout _D avec 259), puis DA (ajout DAD avec 260), etc.

6.2.3. Quel est l'inconvénient de cette méthode, comment y remédier ?

Pour être efficace il faut coder les émissions sur le nombre minimum de bits (8 au début) puis augmenter ce nombre au fur et à mesure des besoins. On peut garder un code 0 pour dire que le nombre de bit est augmenté de 1.

### 6.3. Le codage arithmétique

Le principe de cette méthode est de calculer les fréquences d'apparition des symboles puis d'associer à chaque symbole un intervalle réel compris entre 0 et 1.

Par exemple si on a la chaîne « abbc », on associe les intervalles suivants :

- a : fréquence ¼ : intervalle ]0 ; 0.25[
- b : fréquence ½ : intervalle ]0.25 ; 0.75[
- c : fréquence ¼ : intervalle ]0.75 ; 1[

Par la suite on lit les lettres une par une et on encode :

a : intervalle ]0 ; 0.25[, que l'on découpe en 4 : le premier quart pour a, les deux suivants pour b, le dernier pour c suivant le principe précédent.

b : intervalle ]0.0625 ; 0.1875[ que l'on découpe à nouveau en 4.

b : intervalle ]0.09375 ; 0.15625[, etc.

Une fois terminé le traitement de la chaine, il suffit d'envoyer un réel quelconque dans l'intervalle.

### 6.4. Exercices

6.4.1. Encoder la chaine cbacbbba avec le codage arithmétique.

| | min | max | taille | int0 | int1 | int2 | int3 |
|---|---|---|---|---|---|---|---|
| | | | 1 | 0 | 0,25 | 0,75 | 1 |
| c | 0,75 | 1 | 0,25 | 0,75 | 0,8125 | 0,9375 | 1 |
| b | 0,8125 | 0,9375 | 0,125 | 0,8125 | 0,84375 | 0,90625 | 0,9375 |
| a | 0,8125 | 0,84375 | 0,03125 | 0,8125 | 0,8203125 | 0,8359375 | 0,84375 |
| c | 0,8359375 | 0,84375 | 0,0078125 | 0,8359375 | 0,837890625 | 0,841796875 | 0,84375 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| b | 0,837890625 | 0,841796875 | 0,00390625 | 0,837890625 | 0,838867188 | 0,840820313 | 0,841796875 |
| b | 0,838867188 | 0,840820313 | 0,001953125 | 0,838867188 | 0,839355469 | 0,840332031 | 0,840820313 |
| b | 0,839355469 | 0,840332031 | 0,000976563 | 0,839355469 | 0,839599609 | 0,840087891 | 0,840332031 |
| a | 0,839355469 | 0,839599609 | 0,000244141 | 0,839355469 | 0,839416504 | 0,839538574 | 0,839599609 |

Il suffit de prendre n'importe quelle valeur entre 0.839355469 et 0.839599609

6.4.2. Proposer une méthode de décodage.

De manière symétrique, le nombre est compris entre 0.75 et 1, la première lettre est donc un c, puis il est compris entre 0.8125 et 0.9375 donc la deuxième lettre est un b, etc.

6.4.3. Quel(s) inconvénient(s) voyez-vous à cette méthode ?

Précision en nombre réels sur une machine implique de compresser des chaînes courtes. Nécessité d'avoir la même représentation des flottants sur les différentes machines.