REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MUSTAPHA STAMBOULI DE MASCARA
FACULTÉ DES SCIENCES ET DES TECHNOLOGIES

Polycopié de Cours

CODAGE ET COMPRESSION

Ce cours est destiné aux étudiants de 1ére année Master

Filière : Télécommunications

BESSEGHIER Mokhtar

Algérie
2021

Avant-Propos

- -

Ce support pédagogique met à la disposition des étudiants de télécommunications, un cours sur le codage et la compression des données, visant à les familiariser avec les fondements du codage et la compression des données. Ce cours a pour objectifs donc d'enseigner les différentes techniques du codage de source ainsi que du codage de canal et de présenter leurs applications les plus courantes dans le domaine de la protection et la compression de l'information. Le codage et compression des données sont deux étapes essentielles de tous les systèmes de transmission d'information, la première a pour but de lutter contre les dégradations introduites par la propagation (codage du canal) et la deuxième pour réduire la longueur d'une chaîne sans affecter son contenu informatif (codage de source). Cela permet à la fois de réduire les exigences en mémoire et d'augmenter la capacité d'un canal de transmission (théorie de Shannon).

Pour les compétences requises, l'apprentissage de ce cours « **Codage et Compression** » nécessite des connaissances préalables en communication numérique. En Effet, il est indispensable pour les étudiants de cette matière d'avoir des prérequis concernant les notions suivantes :

- Communication numérique,
- Probabilités et statistiques,
- Théorie d'information,
- Traitement du signal.

Tandis que les compétences acquises, à la suite de cet apprentissage l'étudiant doit être capable de :

- Comprendre les fondements de base pour voir évaluer les avantages et les inconvénients des différentes techniques de compression, ainsi que les différentes techniques de codage canal.
- Connaître les critères de choix d'une technique de compression de données.
- Réaliser un codeur du canal.

Le polycopié est divisé en cinq chapitres. Le chapitre 1 présente des notions fondamentales de codage de source et codage de canal ainsi que le codage conjoint source/canal (CCSC). Ensuite, le deuxième chapitre décrit le codage de source, plus précisément codage de source sans pertes et présente des exemples de ces codes qui existent avec leurs principes de codage et décodage.

Le troisième chapitre décrit les concepts fondamentaux des techniques de codage de canal déployées pour fiabiliser les communications sur les systèmes de transmission. En quatrième chapitre, une description générale sur le codage avec pertes. Il présente donc les différentes méthodes utilisées en compression avec perte, et d'autre part les principaux algorithmes et normes utilisés actuellement pour chaque type de données.

Enfin, dans cinquième chapitre, le cours se termine avec une présentation bien détaillée sur la compression JPEG et une brève présentation sur la compression JPEG 2000.

Le contenu de ce polycopié est enseigné au département d'**Électrotechnique** de la **Faculté des Sciences et de la Technologie à l'Université de Mascara**. Ce cours s'adresse aux étudiants de première année Master filière « Télécommunications » spécialités Réseaux et Télécommunications / Systèmes des Télécommunications.

Le contenu de ce polycopié est conforme au dernier canevas/programme du module **Codage et Compression** recommandé et établit par le Ministère de l'enseignement Supérieur et de la Recherche Scientifique (MESRS).

J'espère que ce modeste travail sera utile et bénéfique à tous ceux qui ont à apprendre ou à enseigner ce module.

Table des matières

Résumé

La transmission des données sous forme numérique doit répondre aux exigences de sécurité, d'efficacité et d'intégrité. Pour y parvenir, les chercheurs s'appuient sur des techniques de codage et compression ces données. Ce cours rassemble l'ensemble des connaissances méthodologiques et algorithmiques usuellement employées en matière de codage et compression d'information. Le chapitre 1 présente des notions fondamentales de codage de source et codage de canal ainsi que le codage conjoint source/canal. Ensuite, le deuxième chapitre décrit le codage de source, plus précisément codage de source sans pertes et présente des exemples de ces codes qui existent avec leurs principes de codage et décodage.

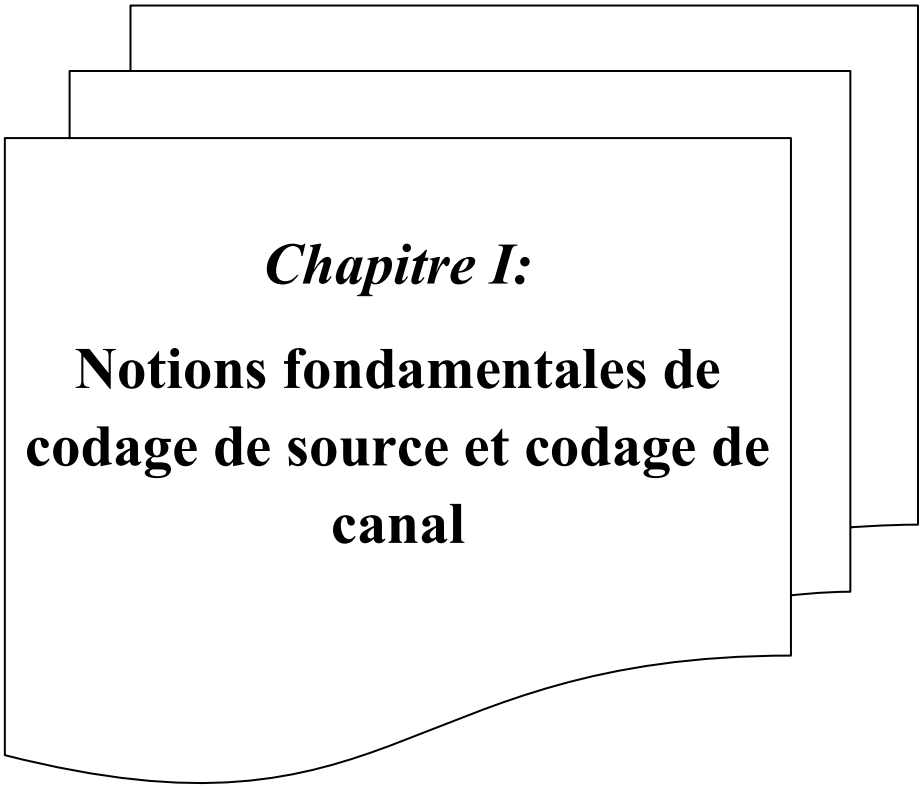
Le troisième chapitre décrit les concepts fondamentaux des techniques de codage de canal déployées pour fiabiliser les communications sur les systèmes de transmission. En quatrième chapitre, une description générale sur le codage avec pertes. Il présente donc les différentes méthodes utilisées en compression avec perte, et d'autre part les principaux algorithmes et normes utilisés actuellement pour chaque type de données. Enfin, dans cinquième chapitre, le cours se termine avec une présentation bien détaillée sur la compression JPEG et une brève présentation sur la compression JPEG2000.

Mots-clés

Information, Codage, Compression, Source, Canal, Entropie, Correction, Probabilité, HUFFMAN, SHANNON, Arithmétique, LZW, Codes en blocs, Codes Convolutifs, Turbo-codes, Image, DCT, DWT, JPEG.

Notations et Abréviations

2D-DCT	Transformée en Cosinus Discrète à deux Dimensions (Bidimensionnelle)
2D-DFT	Transformée de Fourier Discrète à deux Dimensions (Bidimensionnelle)
2D-DWT	Transformée en Ondelettes Discrète à deux Dimensions (Bidimensionnelle)
2D-IDCT	Transformée en Cosinus Discrète Inverse à deux Dimensions (Bidimensionnelle)
2D-IDFT	Transformée de Fourier Discrète Inverse à deux Dimensions (Bidimensionnelle)
2D-IDWT	Transformée en Ondelettes Discrète Inverse à deux Dimensions (Bidimensionnelle)
7z	Seven Zip
ASCII	American Standard Code For Information Interchange
bz2	Bunzip2
CCIR	Comité Consultatif International Pour La Radiodiffusion
CCITT	Comité Consultatif International Télégraphique Et Téléphonique
CCSC	Codage Conjoint Source/Canal
CD	Compact Disc
CDSM	Canal Discret Sans Mémoire
CRC	Cyclic Redundancy Check
CWT	Transformée en Ondelettes Continue
DCT	Transformée en Cosinus Discrète
DFT	Transformée de Fourier Discrète
DIVX	Digital Video Express
DVD	Digital Versatile Disc
DWT	Transformée en Ondelettes Discrète
GIF	Graphics Interchange Format
HDLC	High-Level Data Link Control
i.i.d	Indépendantes et identiquement distribuées
IEC	Commission Électrotechnique Internationale
ISO	International Organization for Standardization
JPEG	Joint Photographic Expert Group
JPEG2000	Joint Photographic Expert Group 2000
KLT	Transformée de Karhunen–Loève
LRC	Longitudinal Redundancy Check
LZ77	Lempel-Ziv 77
LZW	Lempel-Ziv-Welch
MAE	Mean Absolute Error
MP3	MPEG Audio Layer 3
MPEG	Moving Pictures Experts Group
MSE	Mean Squared Error
NSC	Non Systematic Coder
PSNR	Peak Signal-to-Noise Ratio
rar	Roshal ARchive
RGB	Red Green Blue
RLE	Run Length Encoding
RSC	Recursif Systematic Coder
TV	Télévision
SDSM	Source Discrète Sans Mémoire
SNR	Signal-to-Noise Ratio
SSIM	Structural Similarity Index
UIT-T	International Standard Organisation
VLC	Comité Consultatif International Télégraphique et Téléphonique
VRC	Vertical Redundancy Check / Comité Consultatif International pour la Radiodiffusion



Chapitre I:

Notions fondamentales de codage de source et codage de canal

I. Notions fondamentales de codage de source et codage de canal

I.1. Généralités

La théorie des communications traite des moyens de transmettre des informations d'une source à un utilisateur. En général, un système de transmission numérique est représenté par le modèle de base suivant :

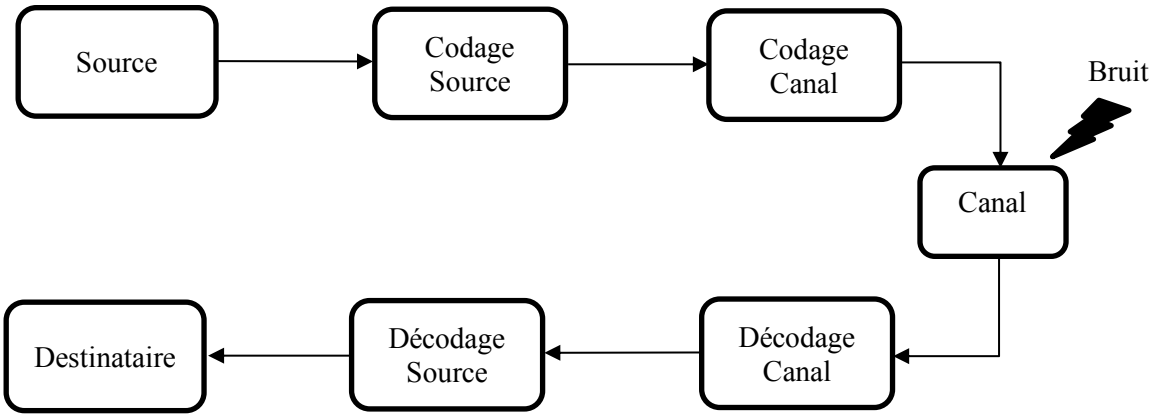


Figure I.1 : Modèle classique d'un système de communication numérique

Dans ce cours, nous nous intéresserons particulièrement aux deux problématiques, l'efficacité et la fiabilité. Et comme cela est connu dans la théorie du codage, une communication efficace entre une source et un destinataire est obtenue grâce au codage de source. Une communication fiable via un canal bruyant est obtenue grâce au codage de canal.

- Le but du codeur de source est de représenter le message avec le moins de bits possible. A cette fin, une tentative est faite pour supprimer toute redondance contenue dans le message de la source.
- Le rôle du codage de canal est de protéger le message contre les perturbations du canal de transmission en ajoutant une redondance au message compressé.

I.2. Sources et codage source

Les sources peuvent être classées en deux catégories :

- Sources analogiques : TV, vidéo, radio (audio général).
- Sources discrètes : disques optiques (CD, DVD, …), les mémoires magnétiques (disques durs, bandes, ...).

I.2.1. Source discrète sans mémoire (SDSM)

La sortie d'une telle source est une séquence de lettres dispose d'un "alphabet" constitué d'éléments ou symboles ou caractères $\{a_1, a_2, a_3, \ldots, a_K\}$. K est la longueur de l'alphabet. Chaque symbole a_k de l'alphabet a une probabilité d'utilisation $P(a_k)$.

On dit qu'une source est sans mémoire si la séquence de symboles générée par la source est une séquence de variables indépendantes et identiquement distribuées (i.i.d).
(Les variables indépendantes et identiquement distribuées sont des variables aléatoires qui suivent toutes la même loi de probabilité et sont indépendantes).

I.2.2. Codage de source

Le codage de source, ou compression des données, est utilisé pour fournir une représentation efficace des données (avec un taux de compression important) tout en préservant les informations essentielles qu'elles contiennent. Les deux types de techniques de codage source sont généralement nommés :

- **Codage sans perte**

La compression (codage de source) est considérée sans perte si les données après la décompression sont identiques aux données originelles (codage réversible). L'objectif principal du codage de source sans perte est donc de supprimer le maximum de redondance pour obtenir une compression plus importante tout en permettant la restauration des redondances supprimées.
Ce type de compression s'applique à tous les types de données et il existe de nombreux formats compressés. Pour ne citer que les plus connus, on retrouve les formats : 7z,bz2, zip, rar, etc...

- **Codage avec perte**

La compression (codage source) est considérée avec perte si les données après la décompression sont différentes aux données originelles (codage irréversible). C'est-à-dire, seule une approximation des données source d'origine peut être reconstruite à partir des données compressées.
Ce type de compression s'appliquent aux données perceptibles, ce qui veut dire aux données multimédia images, sons ou vidéos. Son principe est de supprimer les informations qui ne sont pas perçues par les sens visuel et auditif.
Ici, nous sommes plus susceptibles de parler des techniques du sous-échantillonnage ou la quantification. Parmi ces types : JPEG, MP3, DIVX, MPEG, etc...

Exemple :

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

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

1. *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 » J = « 10 » V = « 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, 01 00 00 01, 01 10 10 10, 01 01 11 00.
   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 29 bits. 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1.
2. *Codage avec perte :* Pour chaque sous bloc 2×2 de l'image on garde la couleur dominante, c'est-à-dire :

R	R
B	J

   Ce qui donne après le codage « trivial » 00 00 01 10 $\Rightarrow$ 8 bits.
   Et après le codage « entropique » 1 1 0 1 0 0 1 $\Rightarrow$ 7 bits.
   Mais bien sûr on ne peut pas revenir à l'image initiale avec ce codage.

- Pour un codage sans perte, on tente de maximiser le taux de compression et de minimiser la complexité de la mise en œuvre ainsi que le retard introduit.
- Pour le codage avec perte, nous essaierons également de maximiser le taux de compression et de réduire la complexité, mais aussi d'obtenir le moins de perte possible.

I.3. Canal et codage de canal

Un canal de transmission reçoit un message entrant et restitue un message sortant. Il peut être vu comme une entité qui crée le lien entre deux alphabets : $X \rightarrow Y$.

I.3.1. Canal de transmission

Le canal est un support physique qui assure la liaison électrique entre l'émetteur et le récepteur.

I.3.2. Canal discret

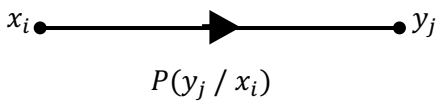
Le canal est dit discret lorsque les deux alphabets d'entrée et de sortie sont des alphabets discrets qui comportent un nombre fini de symboles.

I.3.4. Canal sans mémoire

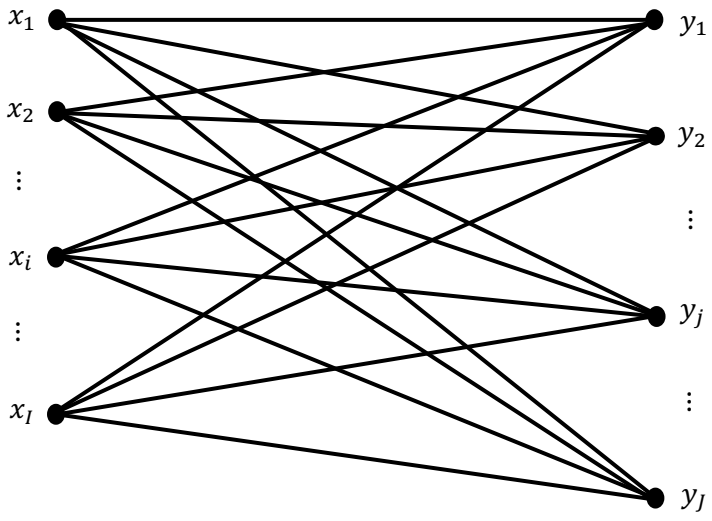
Le canal est dit sans mémoire si le symbole courant de sortie ne dépend que du symbole courant d'entrée et ne dépend ni des précédents ni des suivants.

I.3.5. Représentation graphique

Pour comprendre le fonctionnement du canal discret sans mémoire (CDSM), les données essentielles sont décrites par la probabilité conditionnelle $P(y_k / x_j)$. On peut présenter ces données graphiquement par :



Le schéma global du fonctionnement du canal est donné par le graphe suivant :



Chaque lien est pondéré par la probabilité conditionnelle $P(y_j / x_i)$.

I.3.6. Matrice de canal

Le canal est représenté par la matrice ***P*** des probabilités conditionnelles ou matrice de canal.

$$\boldsymbol{P} = \begin{bmatrix} P(y_1 / x_1) & \cdots & P(y_j / x_1) & \cdots & P(y_J / x_1) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P(y_1 / x_i) & \vdots & P(y_j / x_i) & \vdots & P(y_J / x_i) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ P(y_1 / x_I) & \cdots & P(y_j / x_I) & \cdots & P(y_J / x_I) \end{bmatrix}$$

Première propriété : Nous savons que : $\sum_k P(y_j / x_i) = 1 \Rightarrow$ la somme des éléments d'une ligne est égale à 1.

Deuxième propriété : Rappelons les expressions de la loi de Bayes et des probabilités marginales :

$$P(x_i, y_j) = P(y_j / x_i)\, P(x_i)$$

$$P(y_j) = \sum_i P(x_i, y_j) = \sum_i P(y_j / x_i)\, P(x_i)$$

La dernière expression peut être représenter en fonction de la matrice de canal :

$$\begin{bmatrix} P(y_1) \\ \vdots \\ P(y_J) \end{bmatrix} = \boldsymbol{P}^T \begin{bmatrix} P(x_1) \\ \vdots \\ P(x_I) \end{bmatrix}$$

I.3.7. Codage de Canal

Cela implique d'ajouter une redondance au message entrant et d'exploiter cette redondance en décodant le message reçu pour détecter et corriger éventuelle erreur introduite par le canal.

I.4. Codage conjoint source/canal

Nous avons considéré précédemment, que le codage source (ou compression de données) et le codage canal (correction des erreurs dues au canal) sont indépendant.

Le théorème de codage de source stipule que, tant que les symboles source sont compressés en $L > H$ bits d'information / symbole source, (H: Entropie de la source ou aussi la quantité d'information moyenne de la source) alors la compression de données sans perte est possible.

Le théorème de codage de canal montre que tant que, si les bits d'information $L < C$ (C capacité du canal) alors il existe un codage qui permet d'avoir une transmission avec une probabilité d'erreur de décodage évanescente.

Cela peut indiquer qu'en utilisant une procédure conjointe (la compression des données avec le codage canal), nous pouvons utiliser une source avec entropie H de manière fiable via un canal de capacité C fournie $H < C$.

Question : Serait-il possible de trouver un compromis permettant de combiner les codeurs de source et de canal de façon à réduire la complexité tout en minimisant l'impact sur les performances.

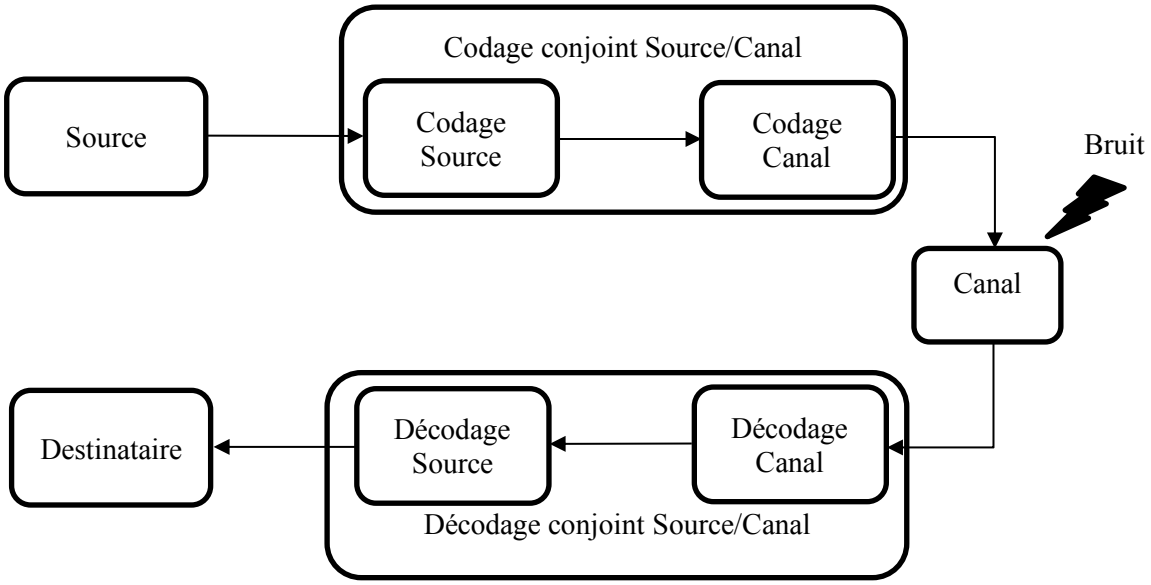
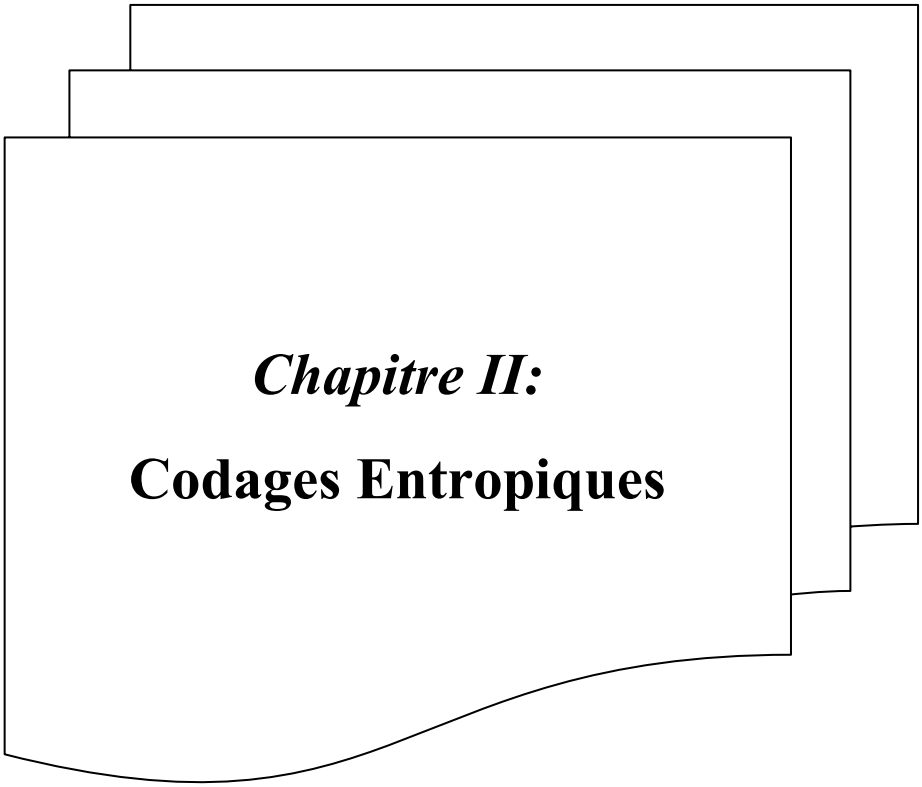


Figure I.2 : Modèle d'un système de communication numérique basé sur CCSC

Dans ce contexte, certains chercheurs ont introduit ce qu'on appelle le codage conjoint source/canal (CCSC), a pour l'objectif de minimiser la distorsion globale en prenant simultanément en compte l'impact des erreurs de transmission et du codage source sur cette distorsion. En général, les techniques du codage conjoint source-canal n'utilisent pas de bits de redondance. Par conséquent, ils n'induisent aucune augmentation du débit de transmission.



Chapitre II:

Codages Entropiques

II. Codages Entropiques

II.1. Généralités

Dans ce chapitre, nous allons nous intéresser au codage de source, plus précisément le codage entropique, appelé également codage statistique à longueur variable, fait partie des méthodes de codage de source sans pertes.

Le codage sans perte nous permet de compresser les données : documents texte, tableurs, fichiers exécutables, …etc. ; il fait souvent partie intégrante de la compression des données lors des transmissions réseau. Parmi les méthodes les plus connues, on trouve le codage RLE, le codage de HUFFMAN, le codage de SHANNON-FANO, le codage arithmétique ou bien encore le codage LZW.

II.2. Mesure De L'information

II.2.1. Quantité d'information

La quantité d'information notée I est une fonction qui satisfait les propriétés suivantes :

1. $I(x_i)$ doit être continue pour $P(x_i)$compris entre *0* et *1*.
2. $I(x_i) \uparrow si\ P(x_i) \downarrow \Rightarrow I(x_i)$ est une fonction décroissante de $P(x_i)$
3. $I(x_i) = \infty\ si\ P(x_i) = 0$ *et* $I(x_i) = 0\ si\ P(x_i) = 1$
4. $I(x_i) + I(y_i) = I(x_i, y_i)$.

Une fonction mathématique remplit les conditions 1, 3 et 4 : est $\log_2(P(x_i))$.

Pour obtenir la propriété 2, il suffit de prendre $-\log_2(p_i) = \log_2(1/P(x_i))$.

La quantité d'information d'un symbole x_i de probabilité $P(x_i)$ a été définie par Shannon comme suit :

$$I(x_i) = -log_2(P(x_i)) = log_2(1/P(x_i)) \qquad (2.1)$$

Où : $\log_2(x) = ln(x) / ln(2)$

Exemple :

Soit une source discrète produisant des bits 0 ou 1 tel que :

- Si $P(0) = P(1) = \frac{1}{2}$.

$$I(x_i) = I(x_i) = -\log_2\left(\frac{1}{2}\right) = 1\ bit$$

- Si $P(0) = 0{,}2$ et $P(1) = 0{,}8$.

$$I(0) = -\log_2(0{,}2) = 2{,}32\ bit$$

$$I(1) = -\log_2(0{,}8) = 0{,}32\ bit$$

Donc, plus l'évènement est peu probable plus il est porteur d'information.

Considérons maintenant la réalisation de 2 évènements $X = x_i$ et $Y = y_j$. La quantité d'information associée est :

$$I(x_i, y_j) = -\log_2(P(\mathrm{x_i}, y_j)) = \log_2(1/P(x_i, y_j)) \tag{2.2}$$

Où $P(x_i, y_j)$ est la probabilité conjointe des deux évènements.

La quantité d'information associée à la réalisation de l'événement $X = x_i$ conditionnellement à l'évènement $Y = y_j$. est la suivante :

$$I(x_i/y_j) = -\log_2(P(\mathrm{x_i}/y_j)) = \log_2(1/P(\mathrm{x_i}/y_j)) \tag{2.3}$$

On en déduit :

$$I(x_i, y_j) = I(x_i/y_j) + I(y_i) = I(y_i/x_j) + I(x_i) \tag{2.4}$$

II.2.2. Entropie d'une source

On détermine l'entropie d'une source décrite par la variable aléatoire X dans un espace de réalisation $A_X = \{x_0, x_1, \ldots, x_{N-1}\}$ avec les probabilités respectives $P_X = \{p_0, p_1, \ldots, p_{N-1}\}$. La quantité d'information moyenne ou l'entropie de la source est la moyenne des informations relatives à chaque réalisation de l'évènement $X = x_i$:

$$\begin{aligned} H(X) &= E[\, I(X)\,] \\ H(X) &= \sum_{i=0}^{N-1} P(x_i) I(x_i) \\ &= \sum_{i=0}^{N-1} P(x_i) \log_2\left(1/_{P(x_i)}\right) \\ &= -\sum_{i=0}^{\mathrm{N}-1} P(x_i) \log_2 P(x_i) \end{aligned}$$

Exprimée en bits/symbole ou Sh/symbole.

Propriétés :

- $H(X) \geq 0$ avec égalité si $p_i = 1$ pour une valeur de i.
- $H(X) \leq \log_2(N)$.
- Si les probabilités $P(x_i)$ sont toutes égales à $1/N$, alors on a : $H(X) = \log_2(N)$.

Par conséquent, l'entropie est maximale lorsque toutes les probabilités $P(x_i)$ sont égales.

Exemple : Soit une source binaire avec deux symboles "0" et "1" de probabilités respectives p et $1 - p$.

$$H(X) = -p \log_2 p - (1 - p) \log_2(1 - p) \tag{2.5}$$

Cette entropie est maximale pour $p = 0{,}5$ et vaut zéro pour $p = 0$ et $p = 1$.

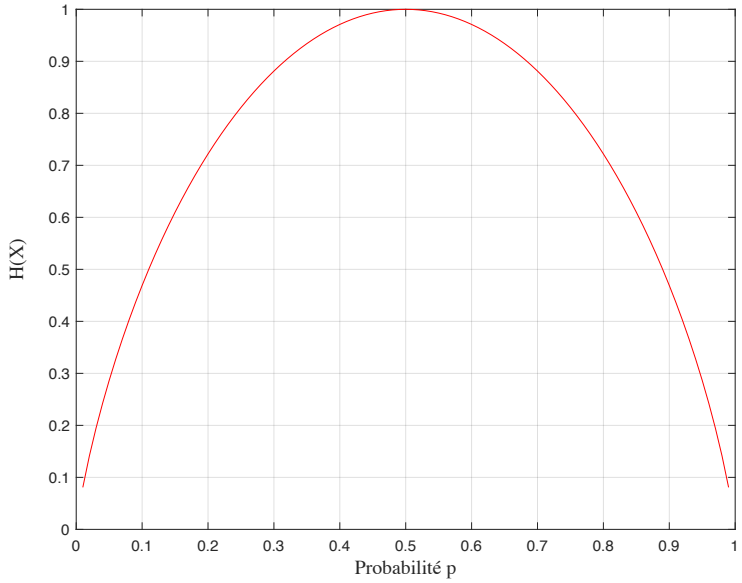


Figure II.1 : Entropie d'une source binaire

II.2.3. Entropie conjointe entre deux sources

Cette notion permet de mesurer l'entropie conjointe $H(X,Y)$ entre deux sources X et Y :

$$X \text{ d'alphabet } \{x_0, x_1, \ldots, x_{N-1}\}$$
$$Y \text{ d'alphabet } \{y_0, y_1, \ldots, y_{N-1}\}$$

L'entropie conjointe des deux sources est alors la quantité d'information moyenne conjointe entre les deux caractères de la source :

$$H(X,Y) = -\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} P(x_i, y_j)\log_2 P(x_i, y_j) \qquad (2.6)$$

Si les variables aléatoires X et Y sont indépendantes alors $H(X,Y) = H(X) + H(Y)$.

II.3. Codage De Source

Le codage de source doit satisfaire les deux critères suivants :

- *Décodage unique* : chaque message devra être codé par un mot différent
- *Déchiffrable unique* : chaque mot de code ne peut être interprétée (décodée) que d'une seule manière

Ce critère s'obtient par :

- ✓ Codage par mot de longueur fixe
- ✓ Codage par mot de longueur variable

Un code est dit *instantané* si aucun mot de code n'est le début d'un autre.

Exemple 1 :

$\mathrm{C} = \{00,01,10,11\}$ est un code de longueur fixe égale à 2.

Exemple 2 :

$\mathrm{C} = \begin{Bmatrix} 1 & 10 & 101 \\ x_0 & x_1 & x_2 \end{Bmatrix}$ est un code de longueur variable mais n'est pas uniquement déchiffrable car : 101 peut être interprété comme $x_1 x_0$ ou x_2.

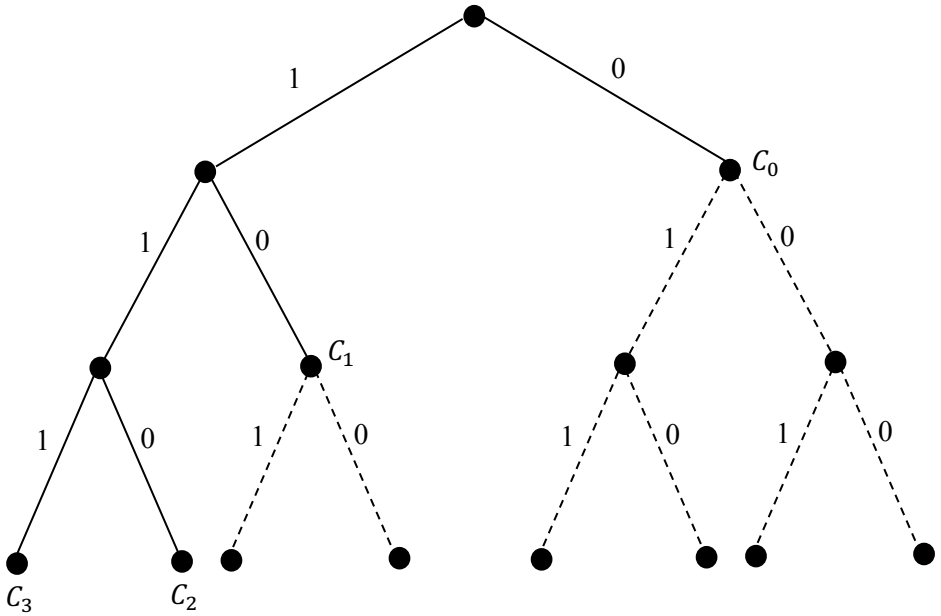
Exemple 3 :

$\mathrm{C} = \{0,10,110,111\}$ est un code de longueur variable.

Un *code préfix* est un code pour lequel aucun mot n'est le début d'un autre. Il est clair que le code préfix est uniquement déchiffrable, également appelé code instantané.

II.3.1. Théorème de Kraft

Un code préfix peut être représenté graphiquement par un arbre binaire complet de profondeur M.



Ce code compose de M mots binaires de longueur respective $\{n_0, n_1, \ldots, n_{M-1}\}$ avec $n_0 \leq n_1 \leq \cdots \leq n_{M-1}$ doit satisfaire l'inégalité suivante :

$$\sum_{i=1}^{M-1} 2^{-n_i} \leq 1 \qquad (2.7)$$

II.3.2. Théorème du codage de source (premier théorème de Shannon)

Soit une source sans mémoire d'entropie par symbole $H(X)$. Il est possible de construire un code préfix dont la longueur moyenne des mots $\bar{L}$ satisfait l'inégalité suivante :

$$H(X) \leq \bar{L} < H(X) + 1 \tag{2.8}$$

Démonstration : Soit $n_i = \lceil -log_2(P(x_i)) \rceil$ longueur du mot associé au $i - ième$ message.

$$\sum_{i=1}^{M-1} 2^{-n_i} = \sum_{i=1}^{M-1} 2^{-\lceil -log_2(P(x_i)) \rceil} \leq \sum_{i=1}^{M-1} 2^{log_2(P(x_i))} = \sum_{i=1}^{M-1} P(x_i) = 1$$

Ainsi on a vérifié que ce code satisfait l'inégalité de Kraft.

On a alors :

$$\bar{L} = \sum_{i=1}^{M-1} P(x_i) \lceil -log_2(P(x_i)) \rceil < \sum_{i=1}^{M-1} P(x_i)\left(-log_2\left(P(x_i)\right) + 1\right) = H(X) + 1$$

II.3.3. Débit D'entropie

Soit une source discrète et $stationnaire^*$ X d'entropie $H(X)$ Sh/symbole délivrant D_s symboles par seconde.

- *Débit d'entropie ou débit informationnel moyen* D_I :

$$D_I = H(X)D_s \text{ en Shannon/seconde} \tag{2.9}$$

- *L'entropie par bit en sortie du codeur de source binaire :*

$$\grave{H}(X) = \frac{H(X)}{\bar{L}} \tag{2.10}$$

- *Le débit binaire* $\grave{D}_b$ *en sortie du codeur :*

$$\grave{D}_b = \bar{L}.D_s \tag{2.11}$$

- *Le débit d'entropie* $\grave{D}_I$ *en sortie du codeur*

$$\grave{D}_I = \grave{H}(X)\grave{D}_b = D_I \tag{2.12}$$

* : La source est dite *stationnaire* si sa loi de probabilité ne dépend pas de l'instant considéré.

II.3.4. Le codage RLE

Run Length Encoding (RLE) est le plus simple des algorithmes de compression de données. Il est basé sur l'élimination des répétitions successives de caractères.

Algorithme *:*

1. Recherche des caractères répétés plus de n fois (n fixé par l'utilisateur).

2. Remplacement de l'itération de caractères par :
    a) Un caractère spécial indiquant la compression.
    b) Le nombre de répétitions du caractère
    c) Le caractère concerné.

Pour la décompression, pendant la lecture du fichier codé, lorsque le caractère spécial est reconnu, l'opération de décodage est effectuée tout en supprimant ce caractère spécial.

Exemple :

Soit le message suivant : TTTTTRRRRRROLLLLBBBBBHHKKKKKK.
On choisit comme caractère spécial : @ et $n = 3$.
Après compression : @5T@6RO@4L@5BHH@6K gain : 11 caractères soit 38%.

II.3.5. Le codage de HUFFMAN

Cet algorithme, basé sur les probabilités des caractères source, est un algorithme optimal qui minimise le nombre moyen de bits utilisés pour le codage et garantit un code uniquement décodable et instantané.

Algorithme *:*

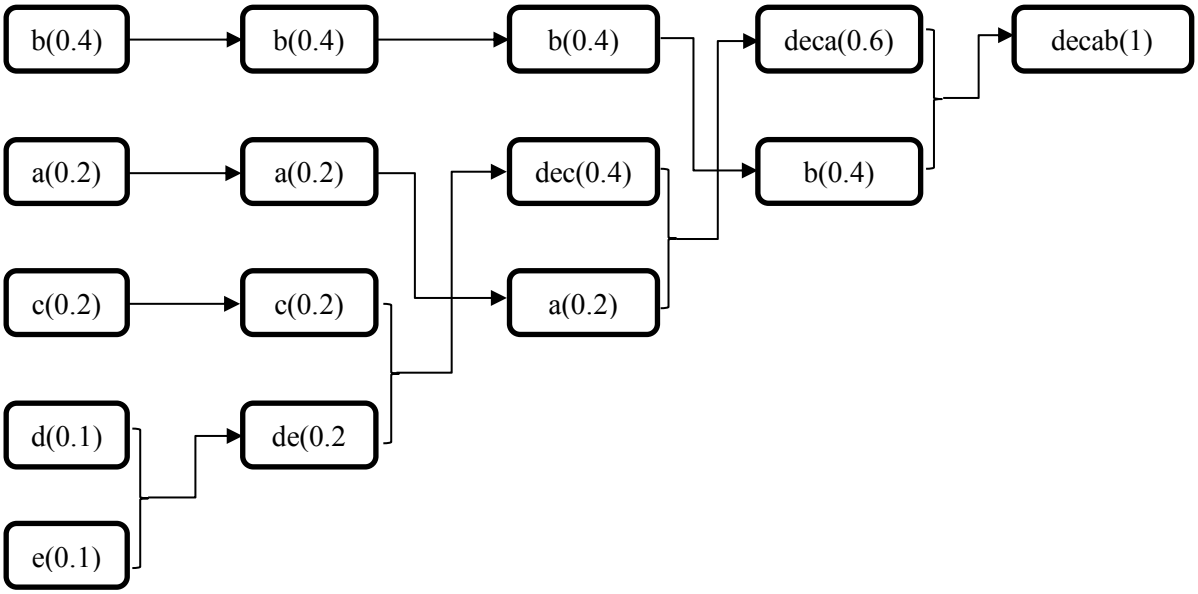
On commence par classer la liste des messages de haut en bas par ordre de probabilité décroissante (chaque message correspond à un nœud).

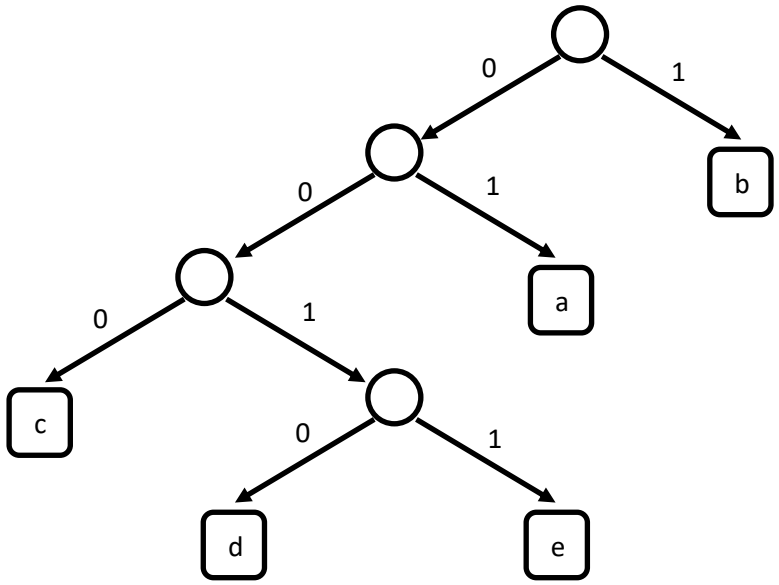
1. Additionner les probabilités des deux messages de probabilité les plus faibles, qui forment un nouvel nœud qui remplace les deux précédents.
2. Nous avons un nouvel ensemble de nœuds avec un nœud en moins et nous reprenons l'étape 1 jusqu'à atteindre à la somme des probabilités totale = 1
3. Pour le codage, on construit un arbre avec retour en arrière, c'est à dire la racine de l'arbre est la dernière itération vers les différents nœuds extrémaux jusqu'aux nœuds de la 1° itération, puis chaque branche de gauche est étiquetée par 0 et la branche de droite par 1.

Exemple :

Soit une source discrète à 5 messages a, b, c, d, e avec les probabilités d'apparition suivantes :

Lettres	a	b	c	d	e
Prob	0.2	0.4	0.2	0.1	0.1





Lettres	a	b	c	d	e
Prob	0.2	0.4	0.2	0.1	0.1
Code	01	1	000	0011	0011
Longueur	2	1	3	4	4

- Longueur moyenne du code : $\bar{L} = 2.2\ bits/symbole$
- Entropie de la source : $H(X) = 2.1219\ bits\ /symbole$
- Redondance du code : $\bar{L} - H(X) = 0.078072\ bits\ /symbole$
- L'efficacité du code est définie par :

$$\eta = \frac{H(X)}{\bar{L}} \tag{2.13}$$

$$\eta = 0.96 = 96\%$$

II.3.6. Le codage de SHANNON-FANO

Le code de Shannon-Fano est le premier code à avoir exploité la redondance d'une source et repose sur la maximisation de l'entropie à la sortie du codeur.

Algorithme :

1. Ordonner les symboles de la source selon des probabilités décroissantes.
2. Séparez les symboles en deux groupes avec des probabilités aussi proches que possible.
3. Le bit de code du groupe supérieur sera "0" et celui du groupe inférieur "1".
4. Répétez cette opération pour chaque sous-ensemble jusqu'à ce que chaque signe possède un code distinct.

Exemple :

Soit une source discrète à 5 messages a, b, c, d, e avec les probabilités d'apparition suivantes :
$P(a) = 0.35, P(b) = 0.22, P(c) = 0.18, P(d) = 0.15, P(e) = 0.1.$

Symbole	$\boldsymbol{P(X)}$	**Étape 1**	**Étape 2**	**Étape 3**	**Code**
a	0.35	0	0		00
b	0.22		1		01
c	0.18	1	0		10
d	0.15		1	0	110
e	0.10			1	111

- Longueur moyenne du code : $\bar{L} = 2.25\ bits/symbole$
- Entropie de la source : $H(X) = 2.19\ bits\ /symbole$
- Redondance du code : $\bar{L} - H(X) = 0.06\ bits\ /symbole$
- L'efficacité du code est définie par :$\eta = 0.97 = 97\%$

II.3.7. Le codage ARITHMETIQUE

Le principe du codage arithmétique repose sur le codage de la séquence des symboles par une valeur fractionnaire de l'intervalle réel [0,1], à travers une procédure itérative encodant un signe à chaque itération.

Ce codage est basé sur la réduction de la largeur de l'intervalle réel, qui dépend de la probabilité que le nouveau symbole soit codé ; la réduction est moindre car le symbole est probable.

Algorithme :

1. A chaque symbole de l'alphabet de la source on associe un intervalle inclus dans [0,1]. La taille de l'intervalle est proportionnelle à la probabilité du symbole.

$$a_i \in [r_i, r_{i+1}[\text{ , avec } r_{i+1} = r_i + P(a_i)\ et r_0 = 0$$

2. Le premier symbole détermine l'intervalle du code (la borne inférieure est choisie).
3. Cet intervalle est ensuite subdivisé en utilisant le même principe (les probabilités) à chaque lecture d'un nouveau symbole et on choisit le sous intervalle du symbole suivant.
4. Le code à la fin peut avoir n'importe quelle valeur dans l'intervalle finale.

Après avoir associer à chaque symbole a_i son intervalle correspondant $[r_i, r_{i+1}[$, l'algorithme suivant permet de coder une chaîne de symboles :

$Inf := 0;$

$Sup := 1;$

$Tans\ qu'il\ y'a\ un\ symbole\ «\ c\ »\ faire:$

$Sup := inf + (Sup - inf) * BornSup[c];$

$inf := inf + (Sup - inf) * BornInf[c];$

$c := LireCarSuivant();$

A la fin la valeur inf est le code du message, ça peut aussi être n'importe quelle valeur dans l'intervalle $[inf, Sup[$.

Exemple :

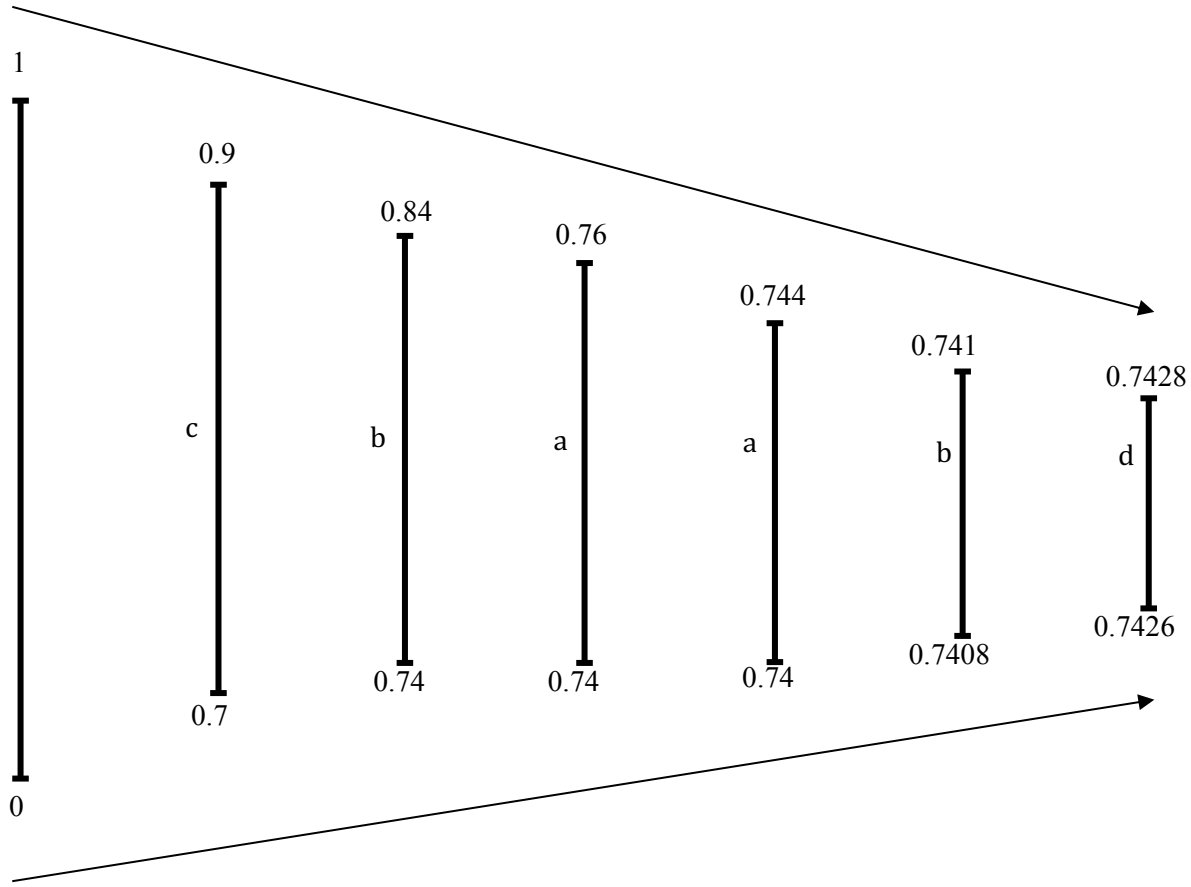
Soit $A = \{a, b, c, d\}$ avec $Prob.\{0.2, 0.5, 0.2, 0.1\}$
On associe donc les intervalles comme suite :

- $a \in [0,0.2[$
- $b \in [0.2, 0.5 + 0.2[= [0.2, 0.7[$
- $c \in [0.7, 0.7 + 0.2[= [0.7, 0.9[$
- $d \in [0.9, 0.9 + 0.1[= [0.9, 1[$

Ainsi pour coder « a » on peut utiliser 0, 0.1, 0.12, 0.1999. Ou n'importe quelle valeur dans [0,0.2[.

Pareil pour b, c et d autant que symboles. Pour coder maintenant un message on applique l'algorithme, par exemple coder : « cbaabd »

- Le premier symbole est « c », on choisit l'intervalle [0.7,0.9[
- Le symbole suivant est «b» donc le nouvel intervalle est: $[0.7 + (0.9 - 0.7) \times 0.2\ ,\ 0.7 + (0.9 - 0.7) \times 0.7\ [= [0.74,0.84[$
- Le symbole suivant est «a» donc le nouvel intervalle est : $[0.74 + (0.84 - 0.74) \times 0\ ,\ 0.74 + (0.84 - 0.74) \times 0.2\ [= [0.74,0.76[$



Symbole	Borne Inf	Borne Sup
c	0.7	0.9
b	0.74	0.84
a	0.74	0.76
a	0.74	0.744
b	0.7408	0.741
d	0.7426	0.7428

Le code de « cbaabd » est donc 0.7426

Le décodage se fait de la manière suivante :

$r := val; message := ''$;

$Répéter \quad \{si\ r \in [r_i, r_{i+1}[\ alors\ message := message + a_i$

$r := (r - r_i)/P(a_i)\ \}\ Jusqu'à\ décodé\ tout\ le\ message$;

En appliquant sur l'exemple on trouve :

$r := 0.7426 \in [0.7,0.9[\rightarrow$ « c »

$r := (0.7426 - 0.7)/0.2 = 0{,}213 \in [0.2,0.7[\rightarrow$ « b »

$r := (0.213 - 0.2)/0.5 = 0{,}026 \in [0,0.2[\rightarrow$ « a »

$r := (0.026 - 0)/0.2 = 0{,}13 \in [0,0.2[\rightarrow$ « a »

$r := (0.13 - 0)/0.2 = 0{,}65 \in [0.2,0.7[\rightarrow$ « b »

$r := (0.65 - 0.2)/0.5 = 0{,}9 \in [0.9,1[\rightarrow$ « d »

La taille du message doit être connue au préalable (sinon on peut boucler à l'infinie)

La valeur réelle obtenu est convertie en binaire avant le stockage : $0.7426 = 1110100000010\ (13\ bit = -\log_2(0.0002))$

Ce qui donne un taux de 2.1 bits par symbole. Qui est très proche de l'entropie du message 1.901.

II.3.8. Le codage de LEMPEL-ZIV-WELCH (LZW)

Cet algorithme a été amélioré par Terry Welch de la société Unisys en 1984 pour l'archivage (les formats ZIP). Il est indépendant des propriétés statistiques de la source et basé sur un dictionnaire (bibliothèque). Le principe de l'algorithme LZW est expliqué sur un exemple comme :

Exemple :

Séquence : somme□somme□somme.

Alphabet $A = \{□, s, o, m, e, .\}$.

Dictionnaire	
Indice	Entrée
1	□
2	s
3	o
4	m
5	e
6	.
Code :	

Dictionnaire	
Indice	Entrée
2	s
3	o
4	m
5	e
6	c
7	so
Code : 2	

Dictionnaire	
Indice	Entrée
3	o
4	m
5	e
6	c
7	so
8	om
Code : 2 3	

Dictionnaire	
Indice	Entrée
4	m
5	e
6	c
7	so
8	om
9	mm
Code : 2 3 4	

Dictionnaire	
Indice	Entrée
5	e
6	c
7	so
8	om
9	mm
10	me
Code : 2 3 4 4	

Dictionnaire	
Indice	Entrée
6	c
7	so
8	om
9	mm
10	me
11	e□
Code : 2 3 4 4 5	

Dictionnaire	
Indice	Entrée
7	so
8	om
9	mm
10	me
11	e□
12	□s
Code : 2 3 4 4 5 1	

Dictionnaire	
Indice	Entrée
8	om
9	mm
10	me
11	e□
12	□s
13	som
Code : 2 3 4 4 5 1 7	

Dictionnaire	
Indice	Entrée
9	mm
10	me
11	e□
12	□s
13	som
14	mme
Code : 2 3 4 4 5 1 7 9	

Dictionnaire	
Indice	Entrée
10	me
11	e□
12	□s
13	som
14	mme
15	e□s
Code : 2 3 4 4 5 1 7 9 11	

Dictionnaire	
Indice	Entrée
11	e□
12	□s
13	som
14	mme
15	e□s
16	somm
Code : 2 3 4 4 5 1 7 9 11 13	

Dictionnaire	
Indice	Entrée
12	□s
13	som
14	mme
15	e□s
16	somm
17	me.
Code : 2 3 4 4 5 1 7 9 11 13 10 6	

Décodage LZW :

Dictionnaire	
Indice	Entrée
1	□
2	s
3	o
4	m
5	e
6	.
7	so
8	om
9	mm
10	me
11	e□
12	□s
13	som
14	mme
15	e□s
16	somm
17	me.

Le code est {2 3 4 4 5 1 7 9 11 13 10 6}, selon le dictionnaire, le message codé est : somme□somme□somme.

II.4. Critères d'évaluation

Le quotient de compression défini par la formule suivante peut être utilisé pour évaluer le degré de réduction des données obtenu par la méthode de compression :

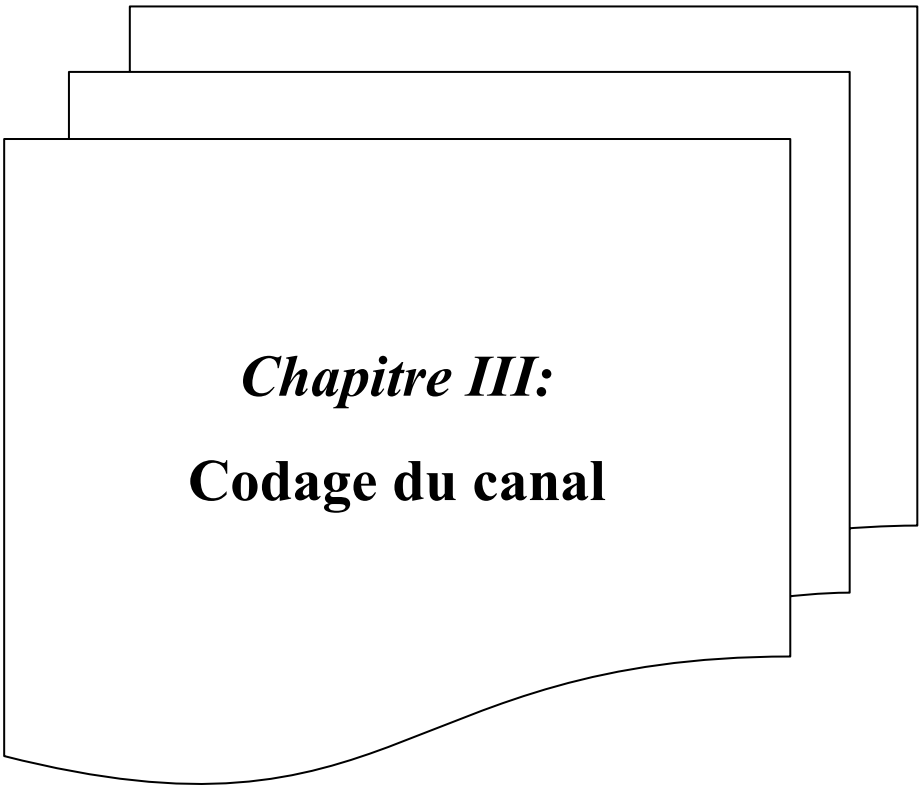
$$Qcomp = \frac{Taille\ initiale}{Taille\ après\ compression} \tag{2.14}$$

Le taux de compression c'est aussi l'inverse du quotient de compression, peut être exprimé en pourcentage :

$$Tcomp = 1/Qcomp \tag{2.15}$$

Le gain de compression c'est aussi le complément à 1 du taux de compression, peut être exprimé en pourcentage :

$$Gcomp = 1 - Tcomp \tag{2.16}$$



Chapitre III:

Codage du canal

III. Codage du canal

III.1. Généralités

Lors du codage source, aucun élément extérieur n'intervient dans la modification des informations. En revanche, lorsqu'on transmet de l'information via un canal bruyant, certains bits reçus seront erronés. La figure suivante représente un dispositif permettant de coder la transmission d'un message sur ce canal.

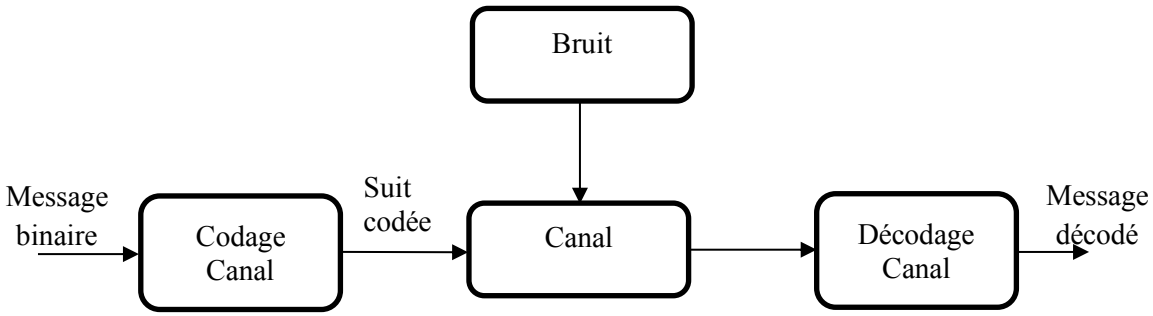


Figure III.1 : Codage d'un canal

III.2. Modèle de transmission

Pour définir un canal de transmission, nous décrivons toutes les entrées et sorties possibles du canal ainsi que le bruit qui peut perturber la transmission. D'un point de vue abstrait nous le considèrerons comme une entité qui établit le lien entre deux alphabets : $X \rightarrow Y$.

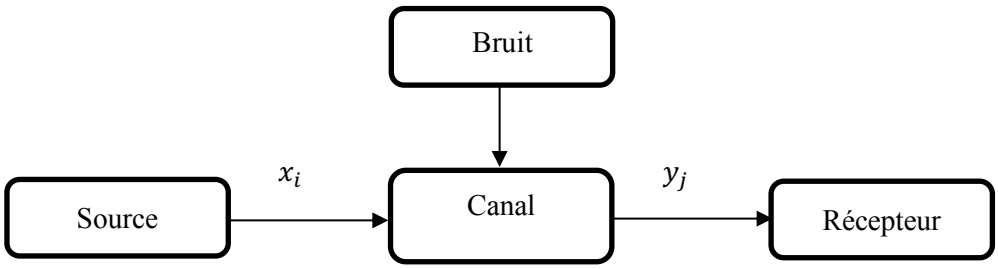


Figure III.2 : Modèle de transmission

- Alphabet de l'émetteur $X = \{x_1, x_2, \dots, x_i, \dots, x_N\}$ $|X| = N$
- Alphabet du récepteur $Y = \{y_1, y_2, \dots, y_j, \dots, y_M\}$ $|Y| = M$

On doit mesure $P(x_i/y_j)$: la probabilité qu'un symbole x_i soit émis sachant que y_j est reçu. Dans le cas idéal on a systématiquement $x_i = y_i$, sinon ceci exprime l'existence d'un bruit.

Le canal est alors modélisé seulement par ça matrice de transition : $P_{ij} = P(Y = y_j/X = x_i)$

- Du point de vue du récepteur :
  - ✓ x est un symbole inconnu.
  - ✓ y est un symbole connu.
  - ✓ x et y sont corrélés (non indépendants)
- La quantité $I(x_i/y_j) = -\log_2\left(P(x_i/y_j)\right) = \log_2\left(1/P(x_i/y_j)\right)$ mesure l'incertitude sur x_i connaissant y_j

Exemple :

- $X = \{a, b\};\ p(a) = p(b) = 1/2\ ;\ I(a) = I(b) = 1\ bit$
- $Y = \{c, d\};\ p(a/c) = p(b/d) = 3/4;\ p(b/c) = p(a/d) = 1/4$

$$I(a/c) = I(b/d) = \log_2(4/3) = (2 - \log_2 3)\ bits$$

$$I(b/c) = I(a/d) = \log_2(4) = 2\ bits$$

  - ✓ $I(a/c) = 2 - \log_2 3 < 1$ donc $I(a/c) < I(a)$ l'incertitude sur « a » a *diminué* en connaissant « c ».
  - ✓ La connaissance de « c » *apporte de l'information* sur « a ».
  - ✓ $I(a/d) = 2 > 1$ donc $I(a/d) > I(a)$ l'incertitude sur « a » a *augmenté* en connaissant « d ».
  - ✓ La connaissance de « d » *apporte de la « désinformation » sur* « a ».

III.3. Information transmise sur un canal

Soit x un symbole émis, et y le symbole reçu en sortie du canal, la quantité :

$$I(x, y) = I(x) - I(x/y) \tag{3.1}$$

Mesure en bit l'information sur x apportée par y.

$$I(x, y) = I(x) - I(x/y) = \log_2\left(1/P(x)\right) - \log_2\left(1/P(x/y)\right) = \log_2\left(P(x/y)/P(x)\right)$$

Exemple :

$$I(a, c) = I(a) - I(a/c) =$$

$$1 - 2 + \log_2 3 = \log_2 3 - 1 > 0$$

« c » apporte une information *positive* sur « a »

$$I(a, d) = I(a) - I(a/d)$$

$$= 1 - 2 = -1 < 0$$

« d » apporte une information *négative* sur « a »

III.3.1. Incertitude sur la source

Pour un symbole y_j reçu, l'incertitude sur la source X par rapport à la réception de y_j est définie par :

$$H(X/y_j) = \sum_{i=0}^{N-1} P(x_i/y_j) \log_2\big(1/P(x_i/y_j)\big) \tag{3.2}$$

On peut montrer que $H(X/y_j) \leq H(X)$: La connaissance d'un symbole reçu diminue toujours l'incertitude sur la source.

En moyenne, l'incertitude sur la source X par rapport à la sortie du canal Y est donnée par :

$$H(X/Y) = \sum_{j=0}^{M-1} P(y_j) H(X/y_j) = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} P(x_i, y_j) \log_2 \left({}^{1}/_{P(x_i/y_j)}\right) \tag{3.3}$$

La valeur de $H(X/Y)$ mesure en bit l'incertitude moyenne sur la source connaissant en moyenne les symboles reçus : l'entropie de la source connaissant la sortie du canal.

III.3.2. Information mutuelle moyenne

L'information mutuelle moyenne définie par :

$$I(X,Y) = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} P(x_i, y_j) I(x_i, y_j)$$
$$I(X,Y) = H(X) - H(X/Y) \tag{3.4}$$

Mesure en bit la quantité d'information apportée par la connaissance de la sortie Y sur la source X. On montre que $I(X,Y) = I(Y,X) \geq 0$. Cette quantité est appelée aussi *Trans-information*

Propriétés :

1) $I(X,Y) = H(X) - H(X/Y)$

$= H(Y) - H(Y/X)$

L'information apportée par la sortie sur la source est égale à l'information apportée par la source sur la sortie (symétrie source/sortie)

2) $0 \leq I(X,Y) \leq H(X)$

$0 \leq I(X,Y) \leq H(Y)$

L'information apportée par Y sur X est au plus égale à l'incertitude sur X ou Y

3) $I(X,Y)\ augment \rightarrow bruit\ diminue \rightarrow Meilleur\ transmission$

$I(X,Y)\ diminue \rightarrow bruit\ augmente \rightarrow Mauvaise\ transmission$

III.4. Capacité d'un canal

Pour tout canal de transmission, nous définissons une grandeur caractéristique appelée capacité du canal, qui peut être interprétée comme la quantité maximale d'informations pouvant passer par le canal.

La capacité d'un canal est définie par l'information mutuelle maximale entre la source X a valeurs sur l'alphabet d'entré du canal, et sa sortie correspondante Y sur le canal.

$$C = \max_{P(X)}(I(X,Y)) \tag{3.5}$$

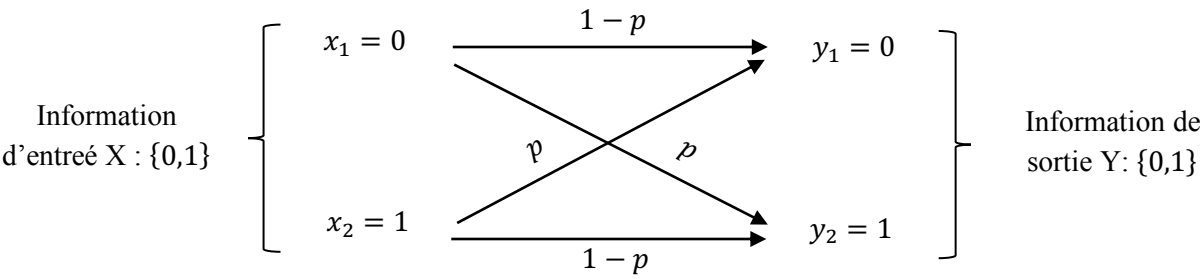
III.4.1. Canal binaire symétrique

Ce modèle de canal binaire sans mémoire est le plus simple. Ses alphabets d'entrée et de sortie sont binaires :

- ✓ $X \rightarrow \{x_1, x_2\}, \{P(x_1) = \alpha, P(x_2) = 1-\alpha\}$
- ✓ $Y \rightarrow \{y_1, y_2\}, \{P(y_1) = ?, P(y_2) = ?\}$

Il est caractérisé par une matrice de canal :

$$\boldsymbol{P} = \{P(y_j/x_i)\} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \tag{3.6}$$



Où p est la probabilité d'erreur de transmission.

Pour calculer la capacité de ce canal il nous faut calculer la quantité d'information mutuelle $I(X,Y)$ entre les deux Informations puis déterminer la valeur de α qui la rend maximale.

$$I(X,Y) = H(Y) - H(Y/X)$$

1. Calcul des $P(y_j)$: La deuxième propriété de la matrice de canal : $[P(y_j)] = \boldsymbol{P}^T\ [P(x_i)]$

$$\begin{bmatrix} P(y_1) \\ P(y_2) \end{bmatrix} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \begin{bmatrix} P(x_1) \\ P(x_2) \end{bmatrix}$$

$$= \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \begin{bmatrix} \alpha \\ 1-\alpha \end{bmatrix}$$

$$= \begin{bmatrix} (1-p)\alpha + p(1-\alpha) \\ p\alpha + (1-p)(1-\alpha) \end{bmatrix}$$

$$= \begin{bmatrix} \alpha + p - 2p\alpha \\ 1 - p - \alpha + 2p\alpha \end{bmatrix}$$

1. Calcul de $H(Y)$:

$$H(Y) = (\alpha + p - 2p\alpha)\log_2 \frac{1}{\alpha + p - 2p\alpha} + (1 - p - \alpha + 2p\alpha)\log_2 \frac{1}{1 - p - \alpha + 2p\alpha}$$

2. Calcul de $H(Y/X)$:

$$\begin{aligned} H(Y/X) &= P(x_1)H(Y/x_1) + P(x_2)H(Y/x_2) \\ &= \alpha\,((1-p)\,log_2 \frac{1}{(1-p)} + p\,log_2 \frac{1}{p}) + (1-\alpha)((1-p)\,log_2 \frac{1}{(1-p)} \\ &\quad + p\,log_2 \frac{1}{p}) \\ &= (1-p)\,log_2 \frac{1}{(1-p)} + p\,log_2 \frac{1}{p} \end{aligned}$$

Donc :

$$I(X,Y) = (\alpha + p - 2p\alpha)\,log_2 \frac{1}{\alpha + p - 2p\alpha} + (1 - p - \alpha + 2p\alpha)\,log_2 \frac{1}{1 - p - \alpha + 2p\alpha}$$
$$- (1-p)\,log_2 \frac{1}{(1-p)} + p\,log_2 \frac{1}{p}$$

3. Finalement, pour maximise $I(X,Y)$ il faut calculer le dérivé par rapport à α:

$$\frac{dI(X,Y)}{d\alpha} = \frac{dH(Y)}{d\alpha} - \frac{dH(Y/X)}{d\alpha} = \frac{dH(Y)}{d\alpha}$$

Comme $H(Y/X)$ ne dépend pas ici de α, il suffit de rendre maximale $H(Y)$.

On pose $\mathrm{f}(\alpha) = \alpha + p - 2p\alpha$ donc $H(Y) = -\mathrm{f}(\alpha)\log_2(\mathrm{f}(\alpha)) - (1 - \alpha)\log_2(1 - \mathrm{f}(\alpha))$

Et on obtient :

$$\begin{aligned} \frac{dH(Y)}{d\alpha} &= \acute{\mathrm{f}}(\alpha)\log_2\left(\frac{1-\mathrm{f}(\alpha)}{\mathrm{f}(\alpha)}\right) \\ &= (1 - 2p)\log_2\left(\frac{1-\mathrm{f}(\alpha)}{\mathrm{f}(\alpha)}\right) \end{aligned}$$

Pour que $I(X,Y)$ soit maximale il faut que : $1 - \mathrm{f}(\alpha) = \mathrm{f}(\alpha) \Rightarrow \mathrm{f}(\alpha) = {}^1/_2 = \alpha - 2p\alpha + p$

$$\Rightarrow \alpha = (1/2 - p)/(1 - 2p) = 1/2.$$

La valeur de $I(X,Y)$ est maximale lorsque les deux symboles de la source d'entrée sont équiprobables $(P(0) = P(1) = 1/2)$ et donc la capacité du canal est donnée par :

$$C = 1 - (1-p)\log_2 \frac{1}{(1-p)} + p\log_2 \frac{1}{p} \qquad (3.7)$$

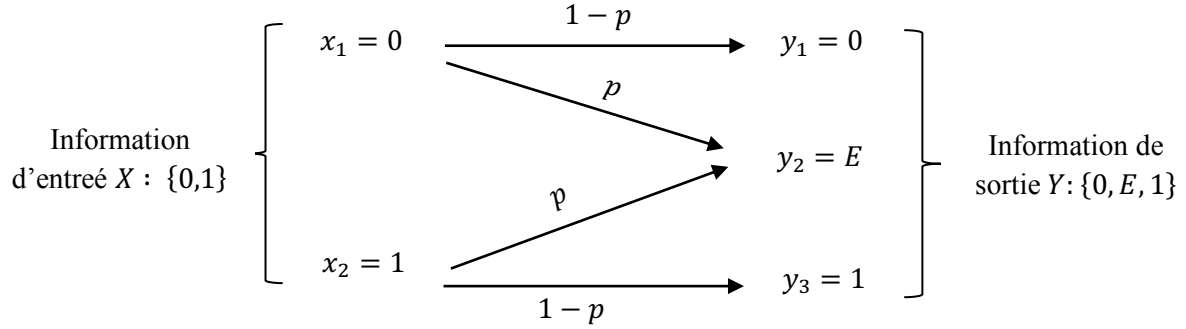
III.4.2. Canal binaire à effacement

C'est un modèle dérivé du canal binaire sans mémoire, dans lequel les bits d'information émis peuvent être perdus ou effaces, ce qui nous donne trois cas à la sortie, le troisième étant l'effacement "E" qui correspond à une erreur de transmission. Ses alphabets d'entrée et de sortie sont :

- ✓ $X \rightarrow \{x_1, x_2\}, \{p(x_1) = \alpha, p(x_2) = 1 - \alpha\}$
- ✓ $Y \rightarrow \{y_1, y_2, y_3\}, \{p(y_1) = ?, p(y_2) = ?, p(y_3) = ?\}$

Il est caractérisé par une matrice de canal :

$$\boldsymbol{P} = \{P(x_i/y_j)\} = \begin{bmatrix} 1-p & p & 0 \\ 0 & p & 1-p \end{bmatrix} \tag{3.8}$$



1. Calcul des $P(y_j)$: La deuxième propriété de la matrice de canal : $[P(y_j)] = \boldsymbol{P}^T [P(x_i)]$

$$\begin{bmatrix} P(y_1) \\ P(y_2) \\ P(y_3) \end{bmatrix} = \begin{bmatrix} 1-p & 0 \\ p & p \\ 0 & 1-p \end{bmatrix} \begin{bmatrix} P(x_1) \\ P(x_2) \end{bmatrix}$$

$$= \begin{bmatrix} 1-p & 0 \\ p & p \\ 0 & 1-p \end{bmatrix} \begin{bmatrix} \alpha \\ 1-\alpha \end{bmatrix}$$

$$= \begin{bmatrix} \alpha(1-p) \\ p \\ (1-\alpha)(1-p) \end{bmatrix}$$

2. Ensuite on calcule $H(Y)$:

$$\begin{aligned} H(Y) &= -\alpha(1-p)\log_2\big(\alpha(1-p)\big) - (1-\alpha)(1-p)\log_2\big((1-\alpha)(1-p)\big) - p\log_2 p \\ &= -\alpha(1-p)(\log_2(\alpha) + \log_2(1-p)) - (1-\alpha)(1-p)(\log_2(1-\alpha) + \log_2(1-p)) \\ &\qquad - p\log_2 p \\ &= -(1-p)(\alpha\log_2(\alpha) + (1-\alpha)\log_2(1-\alpha)) - (1-p)\log_2(1-p) - p\log_2 p \\ &= (1-p)H_2(\alpha) + H_2(p) \end{aligned}$$

3. On calcule ensuite $H(Y/X)$:

$$H(Y/X) = P(x_1)H(Y/x_1) + P(x_2)H(Y/x_2) = P(x_1)H_2(p) + P(x_2)H_2(p) = H_2(p)$$

4. Et donc :

$$I(X,Y) = H(Y) - H(Y/X) = (1-p)H_2(\alpha) + H_2(p) - H_2(p) = (1-p)H_2(\alpha)$$

On dérive par rapport à α :

$$\frac{dI(X,Y)}{d\alpha} = \frac{dH(Y)}{d\alpha} - \frac{dH(Y/X)}{d\alpha} = \frac{dH_2(\alpha)}{d\alpha}$$

$$\frac{dH_2(\alpha)}{d\alpha} = \frac{d(\alpha \log_2(\alpha) + (1-\alpha)\log_2(1-\alpha))}{d\alpha}$$

$$= 1 + \log_2(\alpha) - (1 + \log_2(1-\alpha))$$

$$= \log_2(\alpha) - \log_2(1-\alpha)$$

$$= \log_2\left(\frac{\alpha}{1-\alpha}\right)$$

- Le dérivé s'annule pour $\log_2(\alpha/(1-\alpha)) = 0$ donc $\alpha = 1 - \alpha \rightarrow \alpha = 1/2$ (Distribution uniforme de X).
- La valeur de la capacité est dans ce cas :

$$C = (1-p)H_2(1/2) = 1 - p \tag{3.9}$$

III.5. Deuxième théorème de Shannon

Soit une source discrète sans mémoire (SDSM) X d'entropie $H(X)$ $bits/symbole$ et un canal discret sans mémoire (CDSM) de capacité C $bits/symbole$.

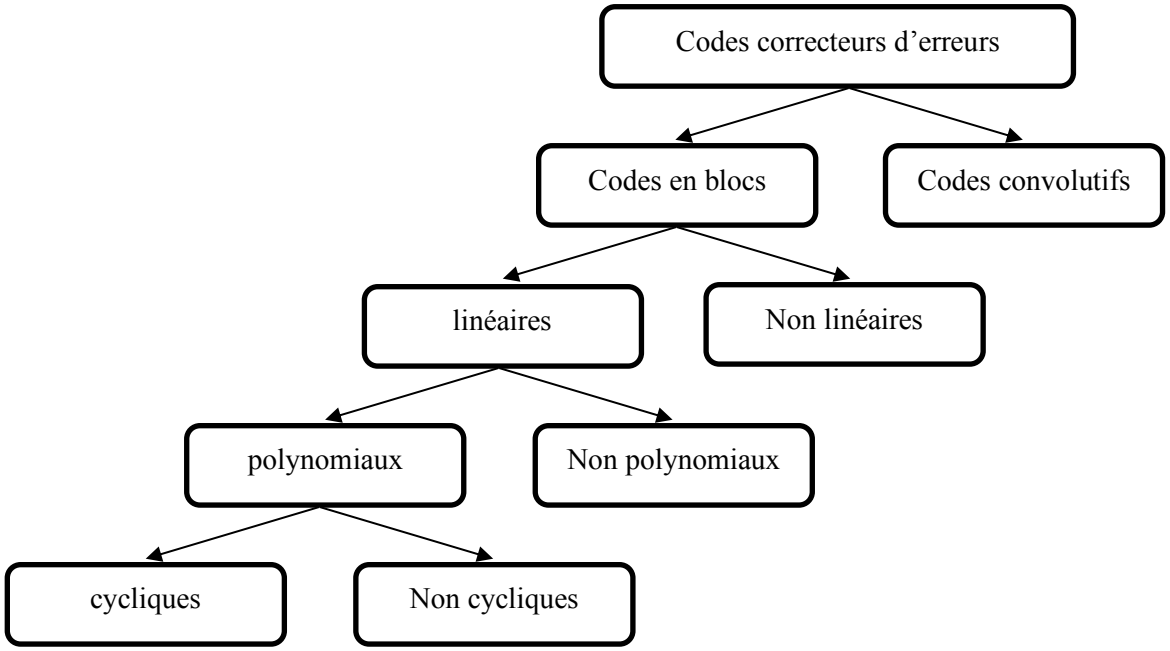
1. Si $H(X) \leq C$, il existe un système de codage tel que l'information de la source peut être transmise sur le canal avec une probabilité d'erreur aussi petite que l'on veut.
2. Si $H(X) > C$, le codage dans ce cas nous permettre d'obtenir une perte d'information proche de $H(X) - C$ et il n'y a pas de méthode de codage qui fournit une perte inférieure` $H(X) - C$.

Ce codage appelé le *codage de canal*.

III.6. Codage d'un canal

L'objectif du codage de canal est de protéger les données du codage source contre les erreurs de transmission. Pour ce faire, nous utiliserons des codes qui ont une structure algébrique telle que la linéarité et rendant ainsi les opérations de codage et de décodage plus simples á effectuer.

On distingue deux grandes familles de codes correcteurs d'erreurs avec la taxonomie suivante :



- **Les codes en bloc** : Ils traitent chaque bloc d'information indépendamment les uns des autres. Chaque mot de code est indépendant des autres mots de code. Pour cela, il consiste à associer à un bloc de mot d'information $\boldsymbol{i}$ de k symboles issus de la source un bloc $\boldsymbol{c}$, appelé un mot de code, de n symboles avec $n \geq k$. La différence $(n-k)$ représente la quantité de redondance introduite par le code.

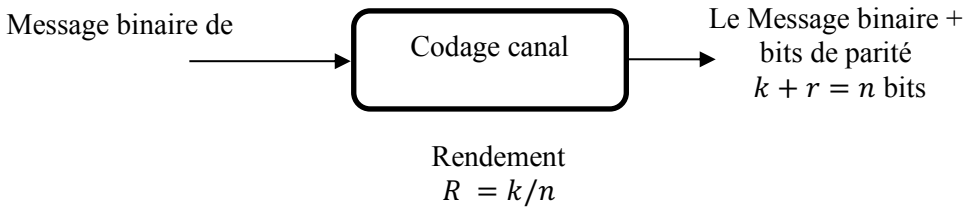
- **Les codes convolutifs** : La sortie d'un codeur convolutif dépend de l'information courante à coder ainsi que de l'information précédente et l'état du codeur. Contrairement aux codes en bloc, Un code convolutif s'applique sur une suite infinie de symboles et produit une suite infinie. On préfère généralement le codage par bloc dans les applications téléinformatiques classiques, le codage/décodage est plus simple.

III.6.1. Propriétés et définitions

Soit $\boldsymbol{i} = [i_0, i_1, \ldots, i_{k-1}]$ un mot d'information composé de k bits d'information et $\boldsymbol{c} = [c_0, c_1, \ldots, c_{n-1}]$ le mot de code associé composé de n bits.

- ***Rendement*** : le rendement R d'un code en bloc (n, k) est égal à :

$$R = \frac{k}{n} \tag{3.10}$$



- ***Distance de Hamming :*** soit $\boldsymbol{x}_1$ et $\boldsymbol{x}_2$ deux mots de code du code $\mathcal{C}$ de longueur n, la distance de Hamming $d_H(\boldsymbol{c}_1, \boldsymbol{c}_2)$ est égale aux nombres de bits qui diffèrent.

  Exemple : $\boldsymbol{c}_1 = [00110010]$ et $\boldsymbol{c}_2 = [00111101]$, $d_H(\boldsymbol{c}_1, \boldsymbol{c}_2) = 4$

- ***Poids de Hamming :*** le poids de Hamming $w(\boldsymbol{c})$ d'un mot de code binaire $\boldsymbol{c}$ est égal au nombre de bits non nuls de ce mot de code.

  Exemple : $\boldsymbol{c} = [0011100]$, $w(\boldsymbol{c}) = 3$.

- ***Distance minimale*** **:** La distance minimale d_{min} du code $\mathcal{C}$ est le nombre de bits qui diffère entre les deux mots de code les plus proches au sens de la distance de Hamming :

$$d_{min} = \min_{i \neq j}\{d_H(\boldsymbol{c}_i, \boldsymbol{c}_j)\} \tag{3.11}$$

Exemple : $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4\}$ avec *:* $\boldsymbol{c}_1 = (00000)$*;* $\boldsymbol{c}_2 = (10110)$*;* $\boldsymbol{c}_3 = (01011)$*;* $\boldsymbol{c}_4 = (11101)$ C'est un code de longueur 5 et de distance $d_{min} = 3$.

- ***Capacité de détection et correction d'erreurs d'un code*** $[n, k, d_{min}]$: Le nombre d'erreurs détectables au maximum est $d_{min} - 1$ et le code de distance minimale d_{min} est corriger

$$e = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor erreurs \tag{3.12}$$

Un bon code doit avoir les propriétés suivantes :

- Un bon *rendement* c'est-à-dire assez de redondance dans les données
- Une bonne *capacité de détection et correction d'erreurs*.
- Une procédure de décodage (et de codage) rapide et moins complexe.

III.6.2. Codage par blocs

Dans un codage par blocs :

- Le message d'information est découpé en blocs de k bits, et le même algorithme est appliqué à chaque bloc.
- Ou nous ajoutons des bits de contrôle à la fin de chaque bloc.
- Ou nous modifions complètement les blocs, mais on évite que deux blocs différents soient transformés en un même bloc.

A) Le code de parité : Généralement, on ajoute 1 bit supplémentaire à la fin de chaque bloc de $n - 1$ bits, valant 1 s'il y a un nombre impair de 1, et 0 sinon. Si à la réception un des n bits est erroné, il y a détection d'erreur.

Exemple :

Lettre	Code ASCII	Mot codé (*parité paire*)	Mot codé (*parité impaire*)
E	1 0 1 0 0 0 1	1 0 1 0 0 0 1 **1**	1 0 1 0 0 0 1 **0**
V	0 1 1 0 1 0 1	0 1 1 0 1 0 1 **0**	0 1 1 0 1 0 1 **1**
A	1 0 0 0 0 0 1	1 0 0 0 0 0 1 **0**	1 0 0 0 0 0 1 **1**

Ce code capable de détecter les erreurs en nombre impair et ne détecte pas les erreurs en nombre pair.

Le code de parité croisée associé d'un double codage de la parité :

	VRC (*parité paire*)
	1 0 1 0 0 0 1 **1**
	0 1 1 0 1 0 1 **0**
	1 0 0 0 0 0 1 **0**
$LRC =$	**0 1 0 0 1 0 1 1**

- *LRC* : "Longitudinal Redundancy Check"

- *VRC* : "Vertical Redundancy Check"

Si le message reçu est :

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

Identifiez l'erreur utilisant la parité croisée

B) Les codes linéaires :

Les codes linéaires sont des codes dans lesquels chaque mot du code $\boldsymbol{c}$ est obtenu par une transformation linéaire des bits du mot d'information $\boldsymbol{i}$. Ces codes sont caractérisés par leur matrice $\boldsymbol{G}(k,n)$ (appelée matrice génératrice), telle que :

$$\boldsymbol{i}.\boldsymbol{G} = \boldsymbol{c} \tag{3.13}$$

Exemple :

$$[1 \quad 0 \quad 1].\begin{bmatrix}1 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\\ 1 & 0 & 0 & 1\end{bmatrix} = [0 \quad 1 \quad 1 \quad 0]$$

- Un code linéaire est dit *systématique*, si sa matrice génératrice possède pour k premières colonnes une matrice identité : $\boldsymbol{G}(k,n) = [\boldsymbol{I}(k), \boldsymbol{P}(k,n-k)]$.
- Alors sa matrice de contrôle s'écrit : $\boldsymbol{H}(n-k,n) = [\boldsymbol{P}^T(n-k,k), \boldsymbol{I}(n-k)]$.

- On peut définir ainsi le *syndrome* $s = \boldsymbol{H}.\boldsymbol{c}^T$, qui est un vecteur-colonne de r bits. Si le syndrome s est nul alors le mot c appartient à l'ensemble des mots du code et sinon, il est égal à la colonne de $\boldsymbol{H}$ correspondant au bit erroné (pour le cas où l'on est sûr de ne pas avoir plus d'une erreur).
- Sous décodage syndromique, un (n, k) bloc de code linéaire peut corriger jusqu'à e erreurs par mot de code si n et k respectent la limite de Hamming.

$$2^{n-k} \geq \sum_{i=0}^{e} C_i^n \tag{3.14}$$

Un codage par blocs qui satisfait à cette égalité est appelé code parfait. Les codes parfaits capables de corriger une seule erreur sont appelés *codes de Hamming*.

Exemple : Trouvez la matrice de contrôle associe à la matrice génératrice suivante

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

La matrice de contrôle $\boldsymbol{H}$ **:**

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

C) Les codes polynômiaux :

Notation : chaque vecteur peut être représenté sous forme polynomiale :

$$1\,1\,0\,1 \rightarrow x^3 + x^2 + 1$$

$$1\,1\,0\,0\,1\,1 \rightarrow x^5 + x^4 + x + 1$$

Définition : Un code polynômial est un code linéaire systématique dans lequel à chacun des mots du code est associé un polynôme divisible par un polynôme générateur noté $\mathrm{g}(x)$.

Propriétés :

- Les opérations sont binaires :
  - ✓ $1.x + 1.x = 0.x$!
- Somme polynômiale :
  - ✓ *Exemple* : $< x^3 + 1 > + < x^4 + x^2 + 1 > = < x^4 + x^3 + x^2 >$
- Produit polynômial :
  - ✓ *Exemple* : $< x^3 + 1 >.< x + 1 > = < x^4 + x^3 + x + 1 >$
- Division polynômiale :
  - ✓ *Exemple* : $< x^4 + x^3 + x^2 + 1 > / < x + 1 > = < x^3 + x + 1 >$
- Modulo :
  - ✓ *Exemple* : $< x^4 + x^3 + x^2 + x + 1 > / < x + 1 > = 1$

Principe du codage **:** Le mot de code $m(x)$ d'un code polynomial (k, n) de polynôme générateur $g(x)$ associé au mot initial $i(x)$ est défini par :

$$m(x) = i(x).x^{n-k} + r(x) \qquad (3.15)$$

Où $r(x)$ est le reste de la division de $i(x).x^{n-k}$ par le polynôme générateur $g(x)$ (noté : $r(x) = i(x).x^{n-k} // g(x)$)

Exemple :

Code polynômial $\mathcal{C}(4,5)$ de polynôme générateur $g(x) = x + 1$

- ✓ $i(x) = x^3 + x + 1$
- ✓ $i(x).x = x^4 + x^2 + x$
- ✓ $r(x) = (x^4 + x^2 + x)//(x + 1) = 1$
- ✓ $m(x) = x^4 + x^2 + x + 1$

Principe du décodage :

A la réception, chaque mot reçu $m'(x)$ est divisé par le polynôme générateur $g(x)$, un reste non-nul indique qu'il y a eu erreur lors de la transmission.

Exemple :

$m'(x) = x^4 + x^2 + x + 1$

$S(m'(x)) = m'(x)//g(x) = (x^4 + x^2 + x + 1)//(x + 1) = 0$

Pas d'erreur détectée

D) Codes Cycliques :

Les codes cycliques sont également des codes polynômiaux dans lesquels les polynômes générateurs sont divisés par $x^n + 1$. Les procédures de codage et de décodage sont les mêmes à ceux des codes polynômiaux. Ce type de code a non seulement la capacité de détecter les erreurs, mais aussi de les corriger.

Exemples de codes polynômiaux :

- ✓ Le polynôme CRC-16 est utilisé par le protocole HDLC :

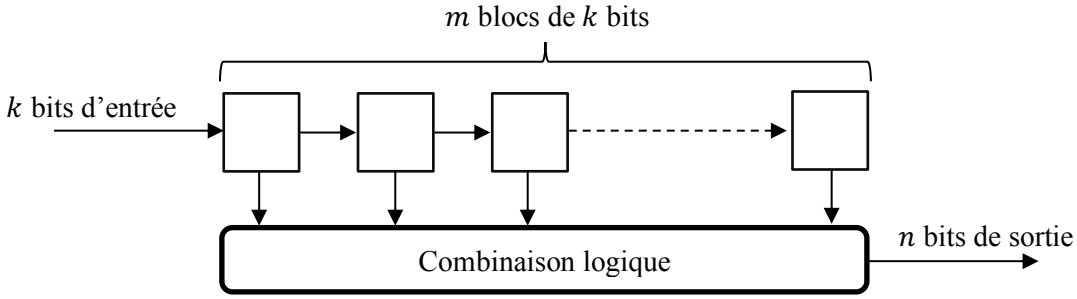
$$g(x) = x^{16} + x^{15} + x^2 + 1$$

- ✓ Le polynôme suivant est utilisé par Ethernet :

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$$

III.6.2. Codes convolutifs

La différence entre les codes convolutifs et les codes en bloc est que chaque bloc de n éléments en sortie ne dépend pas seulement de k entrées à un instant donné, mais aussi de m blocs précédents. Chaque nouveau bloc de k bits d'entrée correspond à un nouveau bloc de n bits de sortie.



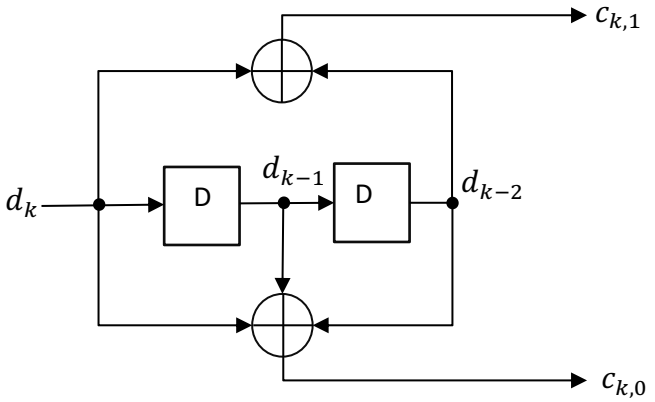
A) Générateur de code :

Considérons un code convolutif avec les paramètres : $m = 2$, $k = 1$, $n = 2$ et $R = 1/2$, comme décrit dans la figure ci-dessous. Chaque sortie du codeur à l'instant k est le produit de la convolution entre la séquence binaire d_k présentée à l'entrée du codeur et la réponse impulsionnelle du codeur définie par ses séquences de génération. Nous avons donc :

$$c_{k,i} = \sum_{j=0}^{m} \mathrm{g_{i,j}} d_{k-j} \tag{3.16}$$

Où j est l'index de la longueur de contrainte et i l'élément du code. Le terme $\mathrm{g_{i,j}}$ est le générateur du code qui spécifie les propriétés du codeur. Dans l'exemple de la figure ci-dessous :

$G_0 = [g_{0,0} \quad g_{0,1} \quad g_{0,2}] = [1 \quad 1 \quad 1] = 7_{(\mathrm{octal})}$ et $G_1 = [\mathrm{g_{1,0}} \quad \mathrm{g_{1,1}} \quad \mathrm{g_{1,2}}] = [1 \quad 0 \quad 1] = 5_{(\mathrm{octal})}$



Donc, on décrit le code convolutif par une matrice génératrice sous la forme suivante :

$$G = \begin{bmatrix} G_0 \\ G_1 \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & g_{0,2} \\ g_{1,0} & g_{1,1} & g_{1,2} \end{bmatrix}$$

Une autre représentation mathématique des codes convolutifs sous forme des polynômes générateurs

$$c_{k,0} = d_k \oplus d_{k-1} \oplus d_{k-2}$$
$$c_{k,1} = d_k \oplus d_{k-2}$$

Si on applique la transformée en Z :

$$C_0(Z) = (1 + Z^{-1} + Z^{-2})\, d(Z)$$
$$C_1(Z) = (1 + Z^{-2})\, d(Z)$$

Où :

$$G_0(Z) = 1 + Z^{-1} + Z^{-2}$$
$$G_1(Z) = 1 + Z^{-2}$$

Nous évoquons une variable D (délai) équivalent à la variable Z^{-1}

$$G_0(D) = 1 + D + D^2$$
$$G_1(D) = 1 + D^2$$

Donc, le code $C(D)$ en fonction de D peut s'exprimer comme :

$$C_0(D) = G_0(D)\, d(D)$$
$$C_1(D) = G_1(D)\, d(D)$$

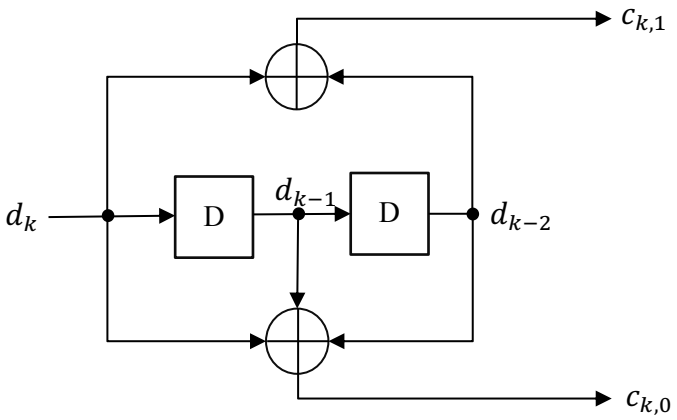
- **Propriétés du codage convolutif**

  - ✓ Le *rendement* du code est : $R = \frac{k}{n}$
  - ✓ La *longueur de contrainte* du code est : $L_c = (m+1)k$
  - ✓ *Linéarité* : les mots de code associés à une combinaison linéaire de séquences d'entrée correspondent à la combinaison linéaire des mots de code de chacune de ces séquences.
  - ✓ *Stationnarité* : Lorsqu'un message source, décalé dans le temps, est envoyé sur l'encodeur, on doit retrouver à la sortie, le mot de code correspondant décalé de la même manière dans le temps.
  - ✓ Code convolutif *systématique* : le mot de code doit contenir les bits d'informations

Les différentes catégories de codes convolutifs sont :

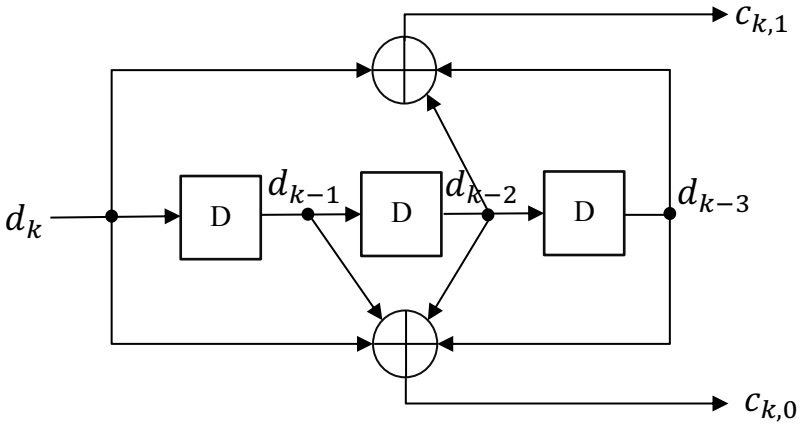
B) Non Systematic Coder (NSC) : ou encore codeur non systématique

Exemple 1 **:** un simple codeur NSC de rendement $R = \frac{1}{2}$ et longueur de contrainte $L_c = 3$



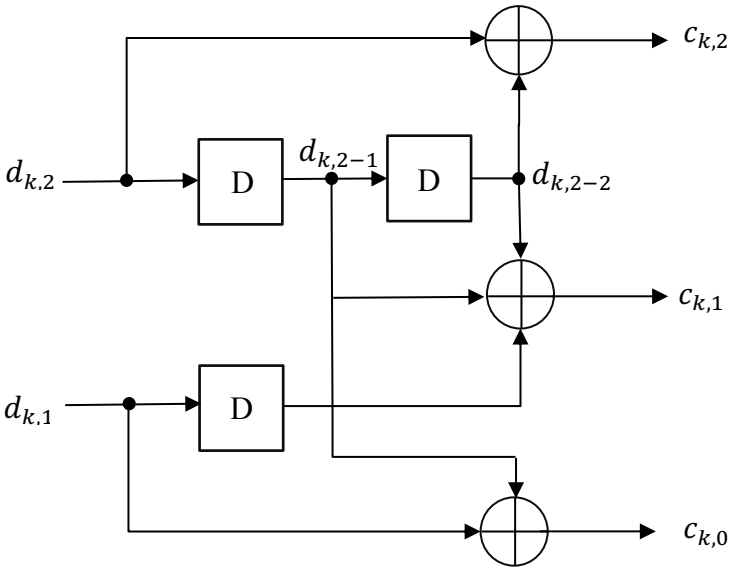
$$G = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Exemple 2 : $k = 1 ; n = 2 ; m = 3$



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

Exemple 3 : $k = 2, n = 3, m = 2$



- Matrices de transfert intermédiaires :

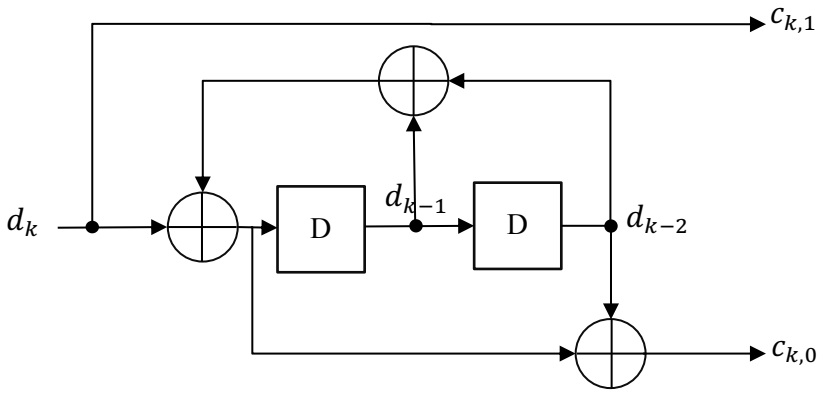
$$G^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$G^1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

$$G^2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

C) Recursif Systematic Coder (RSC) : ou Codeur récursif systématique

Exemple : RSC $(1, 5/7)$



- Polynôme générateur à action directe sous forme binaire est : $G_0 = [1 \quad 0 \quad 1] = 5_{(\text{octal})}$.
- Générateur de rétroaction polynôme sous forme binaire est $G_1 = [1 \quad 1 \quad 1] = 7_{(\text{octal})}$.

La matrice de transfert globale ou génératrice, formée par la concaténation de deux matrices (pour $c_{k,0}$ et $c_{k,1}$) contient l'identité 1, car le codeur est systématique

$$G = [1 \quad G^0/G^1]$$

$$G = \left[1 \quad \frac{1 + D^2}{1 + D + D^2}\right]$$

D) Représentation des codes convolutifs

L'idée de représentation graphique des codes convolutifs vient des caractéristiques de Markov de la sortie du codeur. En effet, la sortie d'un codeur convolutif dépend de son entrée et de ses états. C'est pourquoi les représentations graphiques sont plus faciles à manipuler ce qui nous permet de mieux étudier et modéliser ce type de codeurs. Tout code convolutif peut être représenté par trois graphes équivalents mais différents :

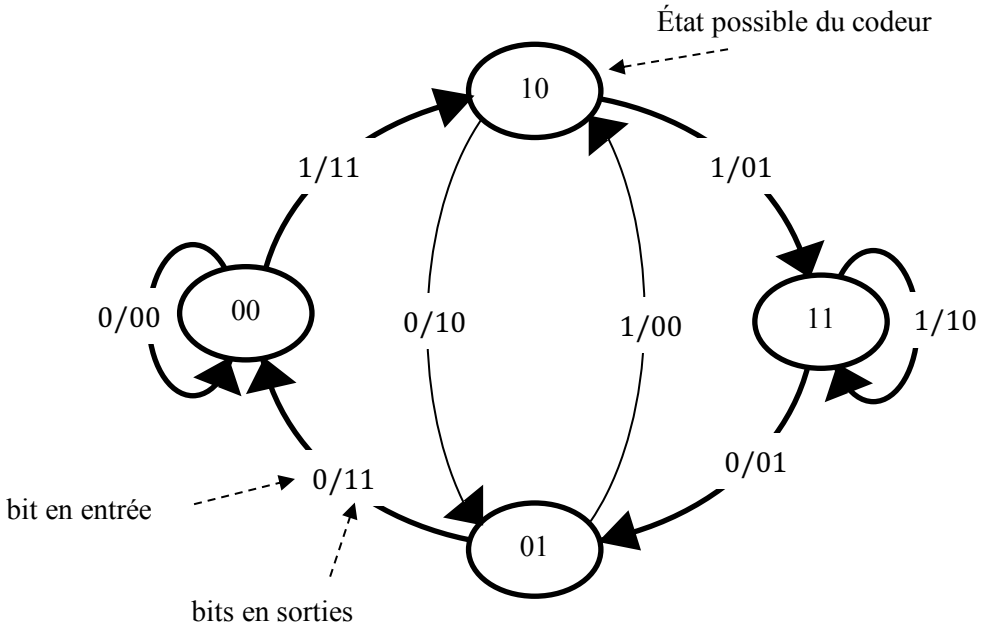
- ✓ Le diagramme d'états,
- ✓ L'arbre du code,
- ✓ Le treillis du code.

- **Diagramme d'états**

Afin de représenter et modéliser du code convolutif, nous pouvons également utiliser des chaînes de Markov, à l'aide du diagramme d'états. Ce dernier, représente les transitions possibles entre les états. Les valeurs des bits sorties $c_{k,0}c_{k,1}$ sont indiqués sur chacune des transitions. Pour un codeur possédant une mémoire de taille m, il existe 2^m états possibles et tous les états internes possibles du codeur sont

représentés par des nœuds S_i. Ou chaque nœud est connecté à un autre via une branche et le passage se fait par une transition : $d_k/c_{k,0}c_{k,1}$.

Pour le cas présenté auparavant (NSC Exemple 1), un tel diagramme est représenté à la figure suivante :

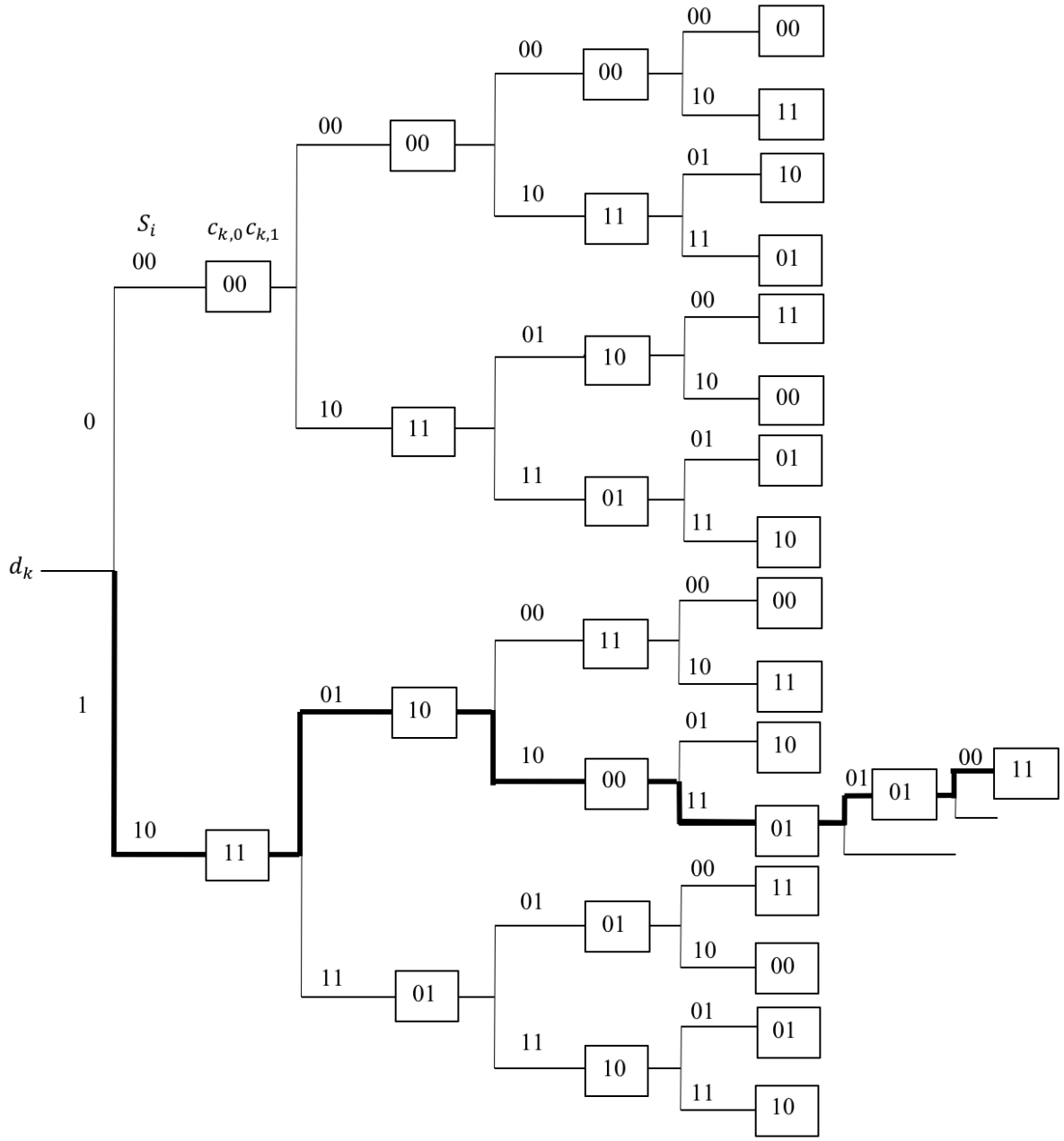


- ✓ Il y'a 2^{L_c-1} états $=$ 4 états.
- ✓ Générateurs : $g_0 = 111, g_1 = 101$
- ✓ Message $= 101100$
- ✓ État initial : $s_0 = 00$

Message codé : 111000010111

- **Diagramme en arbre**

L'arbre est un graphe de hauteur et de largeur infinies. Le sommet représente un état possible du codeur S_i. Il se compose d'arcs et de nœuds. Les arcs sont des traits verticaux dont le sens est déterminé par le bit d'information d_k ... (le 0 est représenté par un arc montant et le 1 par un arc descendant). Les nœuds sont des traits horizontaux indexés par les n sorties $c_{k,0}c_{k,1}$ ... correspondant au bit d'entrée.



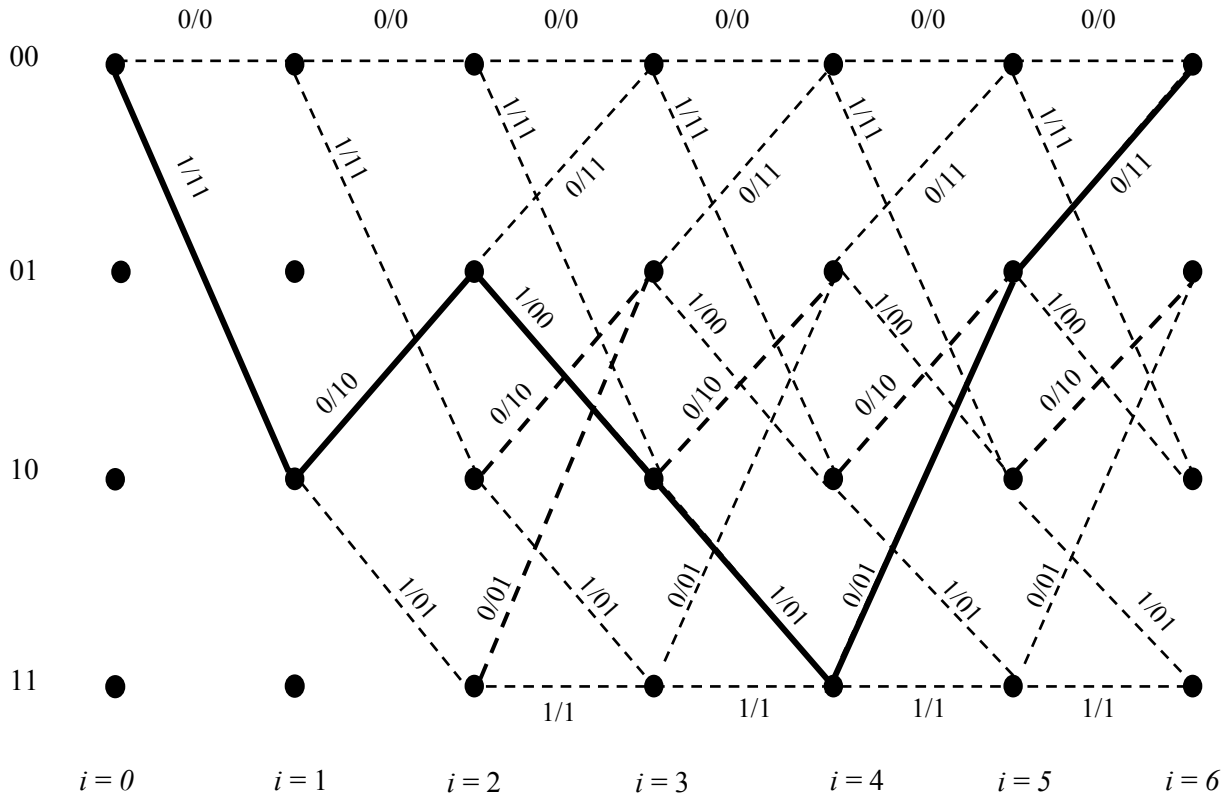
- **Diagramme en treillis**

Dans ce diagramme, on prend en considération les différents états du codeur et la façon dont ils communiquent en fonction du temps. Le treillis est l'outil graphique le plus performant pour représenter et modéliser un code et concevoir des algorithmes de décodage.

La représentation se fait sur deux axes, le premier, l'axe vertical représente les états et le second, l'axe horizontal la variable temps. Les liens qui relient les états à l'instant $i - 1$ et i, indiquent les transitions possibles.

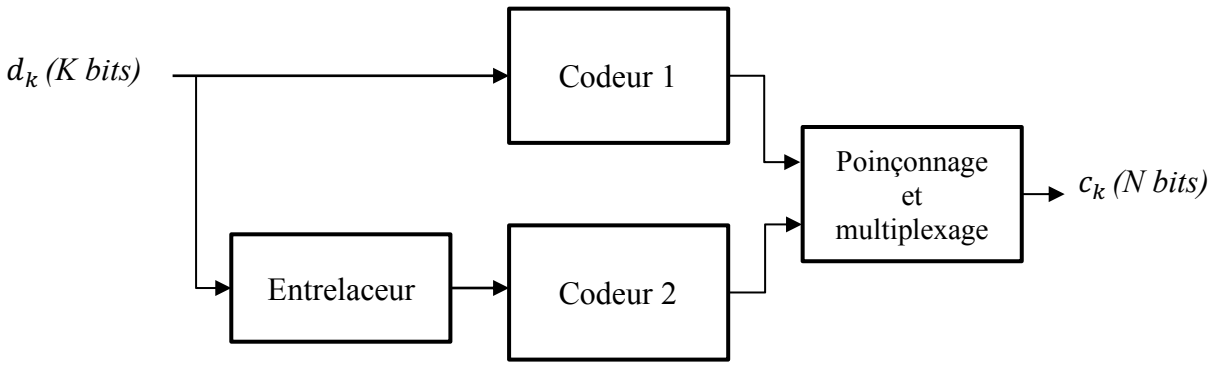
Chaque nœud $S_{(i,j)}$ correspond à un état particulier S_j du codeur à un instant i et chaque branche est indexée par les bits qui se présentent en entrée et en sortie du codeur e/s $(d_k/c_{k,0}c_{k,1})$.

Toujours pour l'exemple que 1'on exploite, on peut trouver une représentation du diagramme en treillis comme suit :



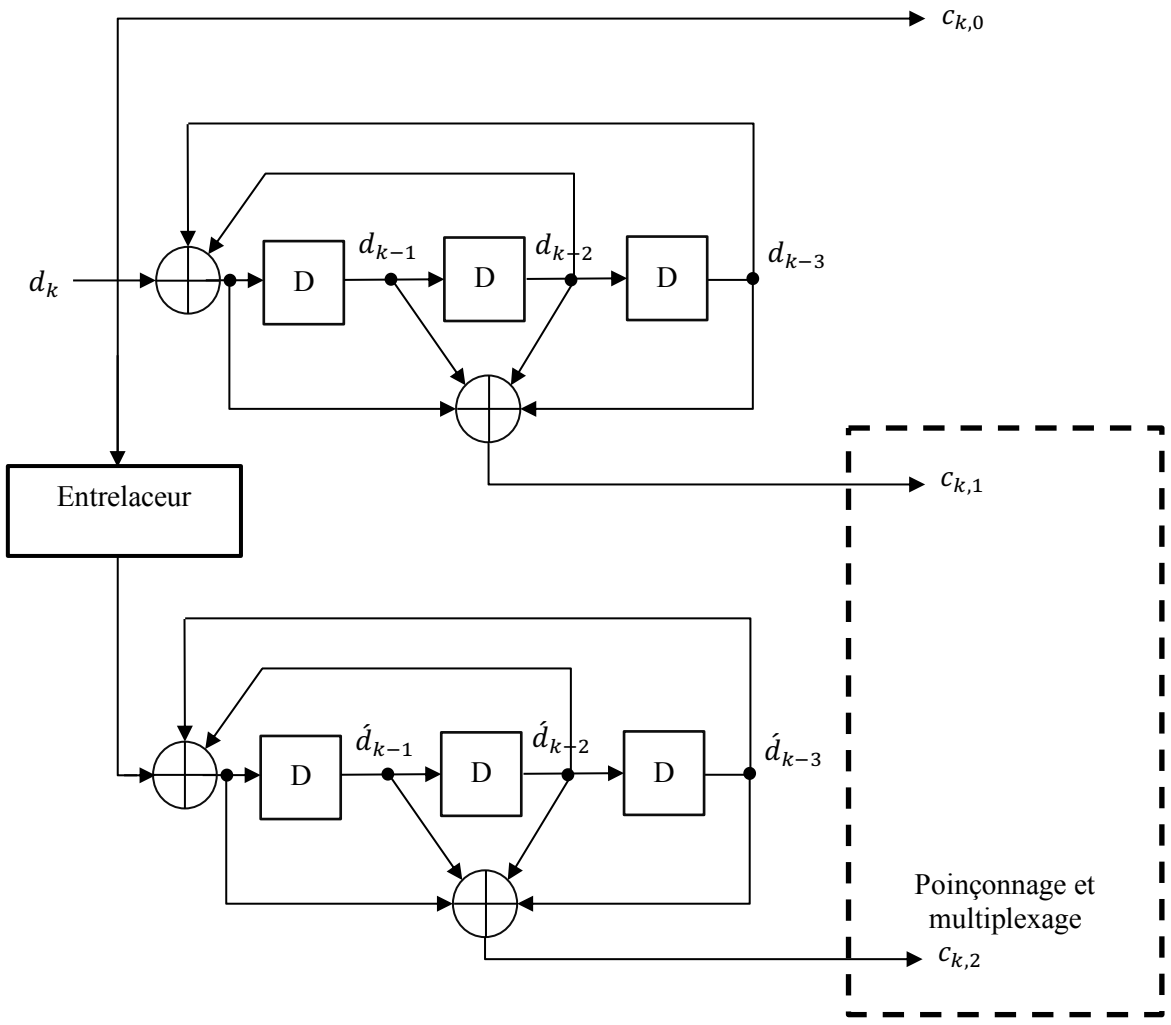
III.7. Turbo-codes

Le principe des turbo-codes est basé sur 1'utilisation de deux petits codeurs convolutifs (concaténés en parallèle), généralement de type récursif systématique (RSC), liés par une fonction de permutation temporelle encore appelée « entrelacement » ou interleavear.

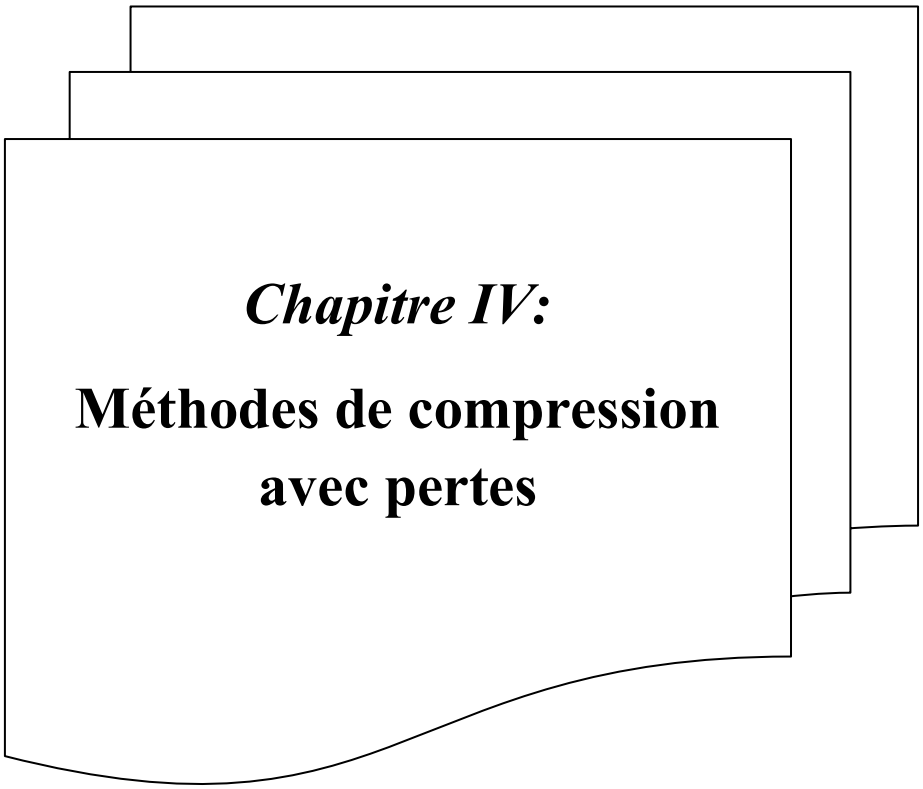


Le poinçonnage, consiste à ne garder à tout instant que l'un des bits envoyés par les deux codeurs, qui peut augmenter le rendement de codage d'un turbo-code. La technique la plus utilisée consiste à poinçonner les symboles transmis suivant un motif périodique.

Exemple : On constate sur le schéma représenté dans la figure suivante que le taux de codage R des turbo-codes est de 1/3 (avant poinçonnage), trois bits de sortie pour un bit d'entrée. On peut le ramener à 1/2 par un poinçonnage qui consiste à ne garder à tout instant que l'un des bits $c_{k,1}$ ou $c_{k,2}$.



Le poinçonnage dans les codes convolutifs est largement utilisé dans la télédiffusion numérique (Digital Video Broadcasting) par satellite.



Chapitre IV:

Méthodes de compression avec pertes

IV. Méthodes de compression avec pertes

IV.1. Généralités

Contrairement aux méthodes de compressions sans pertes, lorsque la transmission d'informations nécessite des débits limités, on s'oriente vers les méthodes tolérant une perte contrôlée de l'information (perte souvent invisible à l'œil ou inaudibles à l'oreille humaine). Ces méthodes dites avec pertes (lossy) permettent d'atteindre des taux de compression élevés pour une qualité acceptable.

IV.2. Compression avec pertes

Ce type de compression comporte une perte de données pendant le processus. Le résultat qu'on peut en obtenir est une version dégradée de la version originale (image ou son). Le but de ce type de compression est d'éliminer le plus d'information possible sans atténuer la qualité de 1' image/son perçue par système visuel/auditif humain. On peut distinguer deux types de compression pouvant générer des pertes : le codage prédictif et le codage par transformée, ce dernier est de loin le plus utilisé dans la compression d'image.

Les méthodes de compression des images fixes, basées sur des transformations linéaires orthogonales, utilisent souvent trois étapes similaires :

- Transformation linéaire (DCT, DFT, KLT, DWT, …).
- Quantification : C'est la seule étape ou il y'à perte.
- Codage entropique.

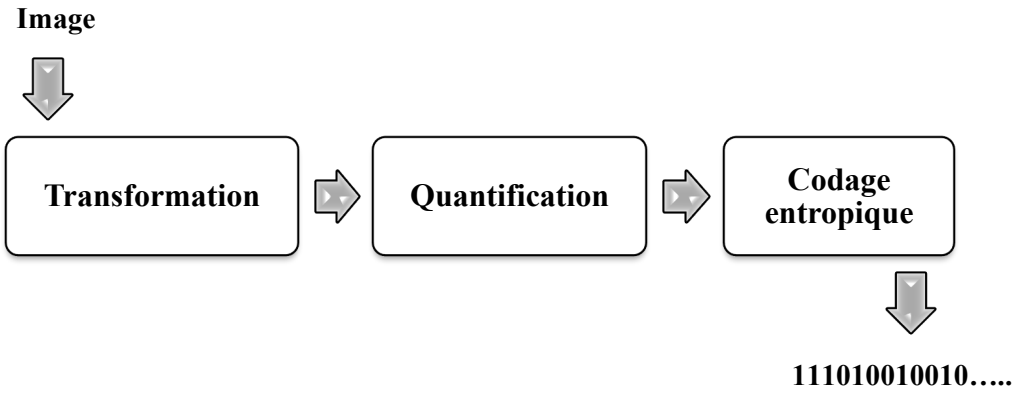


Figure VI.1 : Les différentes étapes de compression des images fixes.

La figure suivante représente le schéma de principe complet d'un codeur/décodeur d'un système de compression avec perte basées sur des transformations.

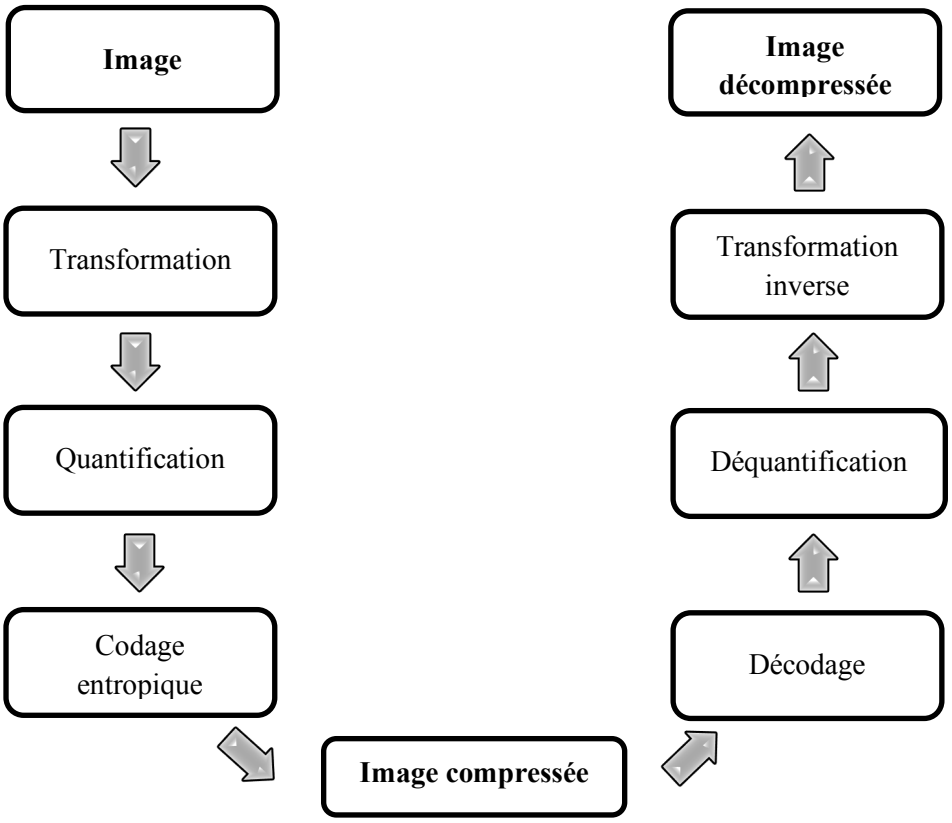


Figure VI.2 : Le schéma de principe d'un système de compression

Premièrement, pour mieux compresser l'information, la source originale est transformée en coefficients. Les transformations les plus utilisées, que ce soit pour les images fixes ou les séquences d'images, sont la Transformée en Cosinus Discrète (DCT) ou la Transformée en Ondelettes Discrète (DWT). Deuxièmement, les coefficients obtenus après la transformation sont quantifiés (tronqués). La phase de quantification introduit l'erreur dans le système de codage. La dernière étape consiste à coder les coefficients quantifiés par le codage entropique.

IV.2.1. Transformée discrète

Les méthodes de compression par transformation n'agissent pas directement sur l'image numérique dans sa représentation canonique, mais sur le domaine de sa transformée. Donc, cette transformation a pour l'objectif :

- Corrélation : obtenir un autre ensemble de valeurs le moins corrélées possible dans l'espace transformé.
- Calcul rapide : convolution vs multiplication.
- Représentation et détection alternatives : obtenir des données transformées comme mesure dans des images de radiologie (médicale et astrophysique).
- Stockage et transmission efficaces.

En général, les algorithmes de compression par transformation subdivisent l'image originale en sous-images de taille plus petite avant d'appliquer à chacune de ces sous images la transformation souhaitée. On peut distinguer plusieurs transformées comme :

- Transformation de Fourrier discrète (TFD).
- Transformation en cosinus discrète (DCT).
- Transformation par ondelette discrète (DWT)

Sous forme discrète ces transformations linéaires représentent un vecteur ou signal $f(n)$ d'un domaine direct et $F(k)$ dans le domaine transformé. Ceci en utilisant un noyau de transformation souvent sous forme d'une base de fonctions orthogonale $\Phi(k,n)$.

Transformée directe

$$F(k) = \sum_{n=0}^{N-1} f(n)\Phi(k,n) \qquad avec : k = 0,1,\dots,N-1 \tag{4.1}$$

Transformée Inverse

$$f(n) = \sum_{k=0}^{N-1} F(k)\Phi^*(k,n) \qquad avec : n = 0,1,\dots,N-1 \tag{4.2}$$

IV.2.2. Transformée de fourrier (DFT)

La transformée de Fourier discrète (TFD) est un outil mathématique pour le traitement du signal numérique. Elle représente la fonction par sa densité spectrale où la variable de l'espace transformé étant la fréquence, une telle décomposition permet de mieux observer la répartition fréquentielle du signal.

Sa définition mathématique pour un signal $f(n)$ de N échantillons est la suivante :

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-j2\pi nk/N} = \sum_{n=0}^{N-1} f(n)w_N^{nk} \tag{4.3}$$

Cette transformée de Fourier peut présenter par la multiplication matricielle avec une matrice qui dépend uniquement de N (Matrice de Vandermonde-Fourier).

$$\boldsymbol{F} = \boldsymbol{W}_N \boldsymbol{f} \tag{4.4}$$

$$\boldsymbol{W}_N = \frac{1}{\sqrt{N}} \begin{bmatrix} w_N^{00} & \cdots & w_N^{0(N-1)} \\ \vdots & \ddots & \vdots \\ w_N^{(N-1)0} & \cdots & w_N^{(N-1)(N-1)} \end{bmatrix}$$

Avec $w_N^{nk} = e^{-j2\pi nk/N}$. La transformée de Fourier Discrète Inverse IDFT :

$$\boldsymbol{f} = \boldsymbol{W}_N^H \boldsymbol{F} \tag{4.5}$$

- **TFD à 2D :**

Considérons une image $N_1 \times N_2$, présenté par la fonction $f(n_1, n_2)$, où $n_1 = 0, \dots, N_1$ et $n_2 = 0, \dots, N_2$. Soit $F(k_1, k_2)$ la transformée de Fourier discrète à deux dimensions (2D-DFT) de l'image. $F(k_1, k_2)$ est donnée par :

$$\begin{aligned} F(k_1, k_2) &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) e^{-j2\pi n_1 k_1 / N_1} \, e^{-j2\pi n_2 k_2 / N_2} \\ &= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) w_{N_1}^{n_1 k_1} w_{N_2}^{n_2 k_2} \end{aligned} \tag{4.6}$$

Avec : $k_1 = 0, \dots, N_1$ et $k_2 = 0, \dots, N_2$. La transformée de Fourier Discrète Inverse à deux dimensions (2D-IDFT) :

$$f(n_1, n_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} F(k_1, k_2) w_{N_1}^{-n_1 k_1} w_{N_2}^{-n_2 k_2} \tag{4.7}$$

IV.2.3. Transformation en Cosinus Discrète (DCT)

C'est une transformation mathématique qui transforme un ensemble de données d'un domaine spatial en un spectre de fréquences, elle est l'une des transformées les plus utilises en compression d'images. En fait, la DCT a une excellente propriété de regroupement d'énergie (l'information est principalement portée par les coefficients basse fréquence).

La DCT est appliquée sur une matrice carrée $N \times N$ de valeurs de pixels et produit une matrice carrée $N \times N$ de coefficients de fréquence. La complexité en temps pour chaque élément dans la DCT dépend de la taille de la matrice. Vu la difficulté d'appliquer la DCT sur la matrice entière, celle-ci est décomposée en blocs de taille 8×8 pixels (compression JPEG).

La DCT unidimensionnelle (1D-DCT) $f(n)$ de N échantillons est donnée par l'équation :

$$F(k) = \sqrt{\frac{2}{N}} \alpha(k) \sum_{n=0}^{N-1} f(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \tag{4.8}$$

Avec $k = 0,1, \dots, N-1$, et $\alpha(k)$ est définit comme :

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}} & pour \ k = 0 \\ 1 & pour \ k \neq 0 \end{cases}$$

La transformée inverse de la 1D-DCT a pour équation :

$$f(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha(k)\, F(k) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \tag{4.9}$$

- **DCT à 2D :**

La DCT bidimensionnelle est une transformation très populaire dans le domaine de la compression d'images fixes, comme le montre son adoption par la norme internationale JPEG. La 2D-DCT d'un bloc de dimension $N \times N$ est donnée par :

$$F(k_1, k_2) = \frac{1}{\sqrt{2N}} \alpha(k_1)\alpha(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} f(n_1, n_2) \cos\left(\frac{\pi(2n_1+1)k_1}{2N}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N}\right) \tag{4.10}$$

Pour calculer la 2D-IDCT :

$$f(n_1, n_2) = \frac{1}{\sqrt{2N}} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \alpha(k_1)\alpha(k_2) F(k_1, k_2) \cos\left(\frac{\pi(2n_1+1)k_1}{2N}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N}\right) \tag{4.11}$$

Pour illustrer la transformation DCT on prend l'exemple suivant :

$$f = \begin{bmatrix} 76 & 76 & 76 & 255 & 255 & 255 & 255 & 255 \\ 76 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 76 & 76 & 149 & 149 & 149 & 255 & 255 & 255 \\ 76 & 255 & 149 & 255 & 149 & 29 & 29 & 29 \\ 76 & 255 & 149 & 255 & 149 & 29 & 255 & 255 \\ 255 & 255 & 149 & 149 & 149 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 29 & 29 \end{bmatrix}$$

Transformation en DCT

$$F = \begin{bmatrix} 1474 & -3 & -38 & -128 & 49 & -8 & -141 & 11 \\ 1 & -171 & -51 & 35 & -45 & -4 & 40 & -58 \\ 93 & -2 & -43 & 45 & 26 & 0 & 38 & 24 \\ 61 & -66 & 70 & -11 & 23 & 15 & -48 & 6 \\ -113 & 62 & -71 & -3 & 27 & -10 & -25 & -33 \\ -47 & 1 & -5 & 18 & 28 & 14 & 4 & 2 \\ -72 & -37 & 37 & 51 & 7 & 18 & 33 & -1 \\ 41 & -100 & 21 & -9 & 2 & 27 & 7 & 7 \end{bmatrix}$$

IV.2.4. Transformation en ondelettes (Wavelet Transforms : WT)

La transformée en ondelettes est similaire à la transformée de Fourier. La différence est que la taille de la fenêtre utilisée par TF est fixe, par contre la WT utilise des fenêtres de taille dépendant de la fréquence analysée mais avec un nombre d'oscillations fixe.

La transformée en ondelettes d'un signal $f(t)$ est donc :

$$W(a,b) = \int_{-\infty}^{+\infty} f(t)\,\psi_{a,b}(t)\,dt \tag{4.12}$$

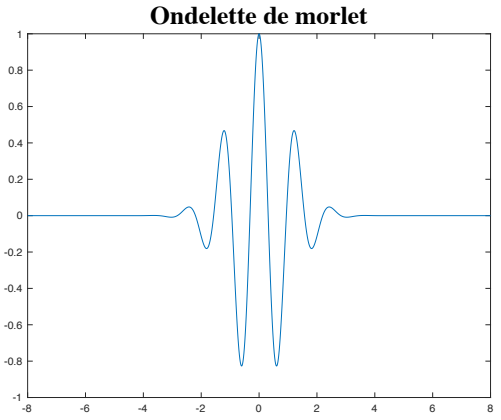
L'ondelette $\psi_{a,b}(t)$ est définie par la formule :

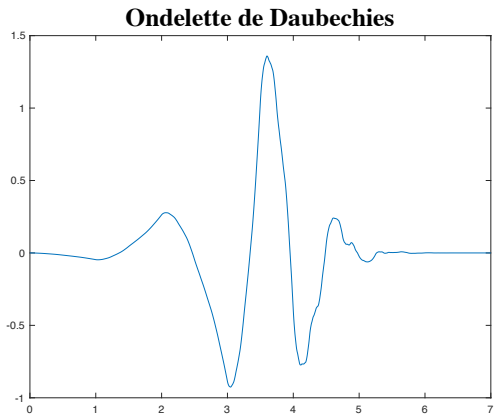
$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \tag{4.13}$$

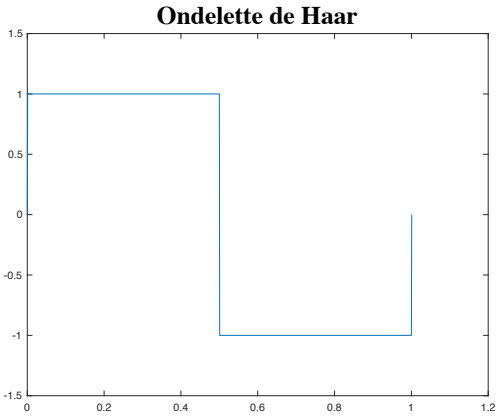
Où a et b sont deux nombres réels.

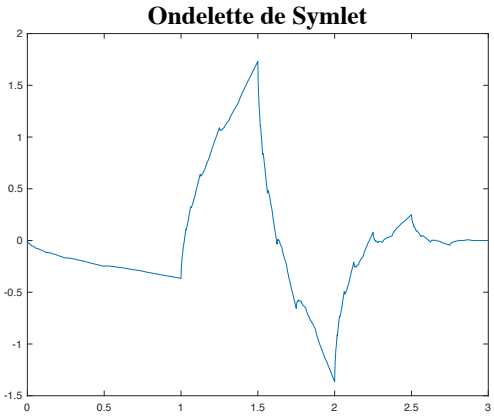
- « a » est le paramètre d'échelle qui joue le rôle de la fréquence dans la transformée de Fourier à fenêtre (a petit => des fréquences élevées et inversement).
- « b » est le paramètre de translation qui joue le rôle de la position de la fenêtre dans la transformée de Fourier à fenêtre correspondant donc à l'axe des temps.

Exemples d'ondelettes :









$W(a,b)$ est appelée la transformée en ondelettes continue (CWT), où a et b sont des variables continues et $f(t)$ est une fonction continue. La transformée en ondelettes inverse est obtenue comme suit :

$$f(t) = \frac{1}{C} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} W(a,b)\, \psi_{a,b}(t)\, da\, db \tag{4.14}$$

Avec :

$$C = \int_{-\infty}^{+\infty} \frac{|\Psi(\omega)|^2}{\omega}\, d\omega$$

Où : $\Psi(\omega)$ est la transformée de Fourier de $\psi(t)$. La transformée en ondelettes continue inverse existe si le paramètre C est positif et fini.

Dans les applications pratiques, les paramètres a et b sont discrétisés, et pour la discrétisation du paramètre a, on choisit les puissances du paramètre de dilatation fixe $a_0 > 1$:

$$a = a_0^j \qquad m \in \boldsymbol{Z}$$

Et pour b, nous utilisons :

$$b = kb_0a_0^j \qquad m \in \boldsymbol{Z}, b_0 > 0$$

En utilisant les paramètres discrets a et b, nous obtenons la famille discrétisée d'ondelettes :

$$\psi_{a,b}(t) = a_0^{-j/2}\, \psi\left(a_0^{-j}t - kb_0\right)$$

Pour $a_0 = 2$ et $b_0 = 1$, donc $a = 2^{-j}$ et $b = k2^{-j}$

On obtient alors :

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k)$$

Le WT a pour objectif de décomposer le signal en deux parties de coefficients à l'aide de filtres passe-haut et passe-bas. Les coefficients de la transformation en ondelettes discrètes (DWT) sont définis par les relations suivantes :

$$W_{DC}(j,k) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f(i)\psi_{j,k}(i) \tag{4.15}$$

$$W_{AC}(j_0,k) = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f(i)\varphi_{j_0,k}(i) \tag{4.16}$$

Avec $W_{AC}(j_0,k)$ est appelé le coefficient d'approximation à l'échelle j_0 et $W_{DC}(j,k)$ est appelé les

coefficients de détail à l'échelle j où : $j \geq j_0$.

Les coefficients DWT nous permettent de reconstruire la fonction de signal $f(i)$ exprimée par représentation suivante :

$$f(i) = \frac{1}{\sqrt{N}}\left(\sum_{k=-\infty}^{+\infty} W_{AC}(j_0,k)\,\varphi_{j_0,k}(i) + \sum_{j=j_0}^{M-1}\sum_{k=-\infty}^{+\infty} W_{DC}(j,k)\,\psi_{j,k}(i)\right) \tag{4.17}$$

Où : $\psi(i)$ est l'ondelette mère et $\varphi(i)$ est la fonction de mise à l'échelle.

Les coefficients $W_{AC}(j_0,k)$ et $W_{DC}(j,k)$ peuvent être calculé récursivement par :

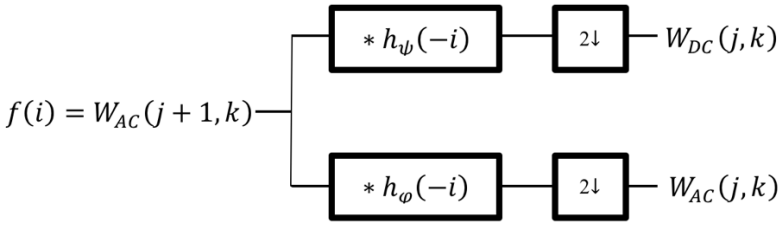
$$W_{AC}(j,k) = \sum_i h_\varphi(i-2k)\,W_{AC}(j+1,k) \tag{4.18}$$

$$W_{DC}(j,k) = \sum_i h_\psi(i-2k)\,W_{DC}(j+1,k) \tag{4.19}$$

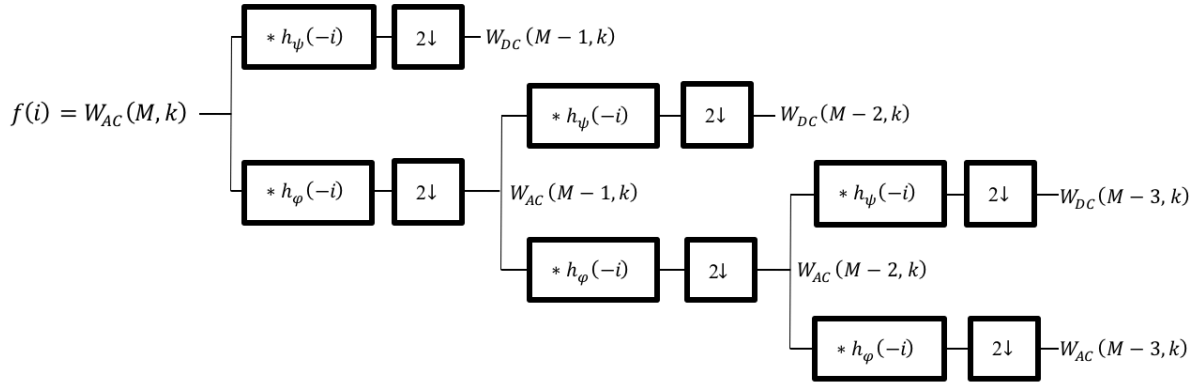
$$W_{AC}(j,k) = W_{AC}(j+1,k) * h_\varphi(-i)$$

$$W_{DC}(j,k) = W_{DC}(j+1,k) * h_\psi(-i)$$

Où $h_\varphi(i) = \langle \varphi(x), \sqrt{2}\,\varphi(2x-i)\rangle$ ($\langle f(x), \mathrm{g}(x)\rangle = \int_a^b f^*(x)\,\mathrm{g}(x)\,dx$) appelé les coefficients de fonction d'échelle et $h_\psi(i) = (-1)^k h_\varphi(1-i)$ appelé les coefficients de la fonction d'ondelettes, où les convolutions sont évaluées à des instants $i = 0, 2, \ldots, 2^j - 2$. Comme indiqué sur la figure suivante :



Un exemple de décomposition au niveau 3 d'un signal est représenté sur la figure suivante :



- **DWT à 2D :**

Pour la transformée en ondelettes bidimensionnelle (2D-DWT), une fonction de mise à l'échelle de deux dimensions $\varphi(x,y)$ et trois ondelettes de deux dimensions, $\psi^H(x,y)$, $\psi^D(x,y)$ et $\psi^V(x,y)$, sont nécessaires.

Où :

- ✓ $\varphi(x,y) = \varphi(x)\varphi(y)$
- ✓ $\psi^H(x,y) = \psi(x)\varphi(y)$
- ✓ $\psi^D(x,y) = \psi(x)\psi(y)$
- ✓ $\psi^V(x,y) = \varphi(x)\psi(y)$

$$\varphi_{j,m,n}(t) = 2^{-j/2}\varphi(2^{-j}x - m, 2^{-j}y - n)$$

$$\psi^i_{j,m,n}(t) = 2^{-j/2}\psi^i(2^{-j}x - m, 2^{-j}y - n)$$

Avec : $i = \{H, V, D\}$

La transformée en ondelettes discrète de la fonction $f(x,y)$ de taille $M \times N$ est alors :

$$W_{AC}(j_0,m,n) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(x,y)\, \varphi_{j_0,m,n}(x,y) \tag{4.20}$$

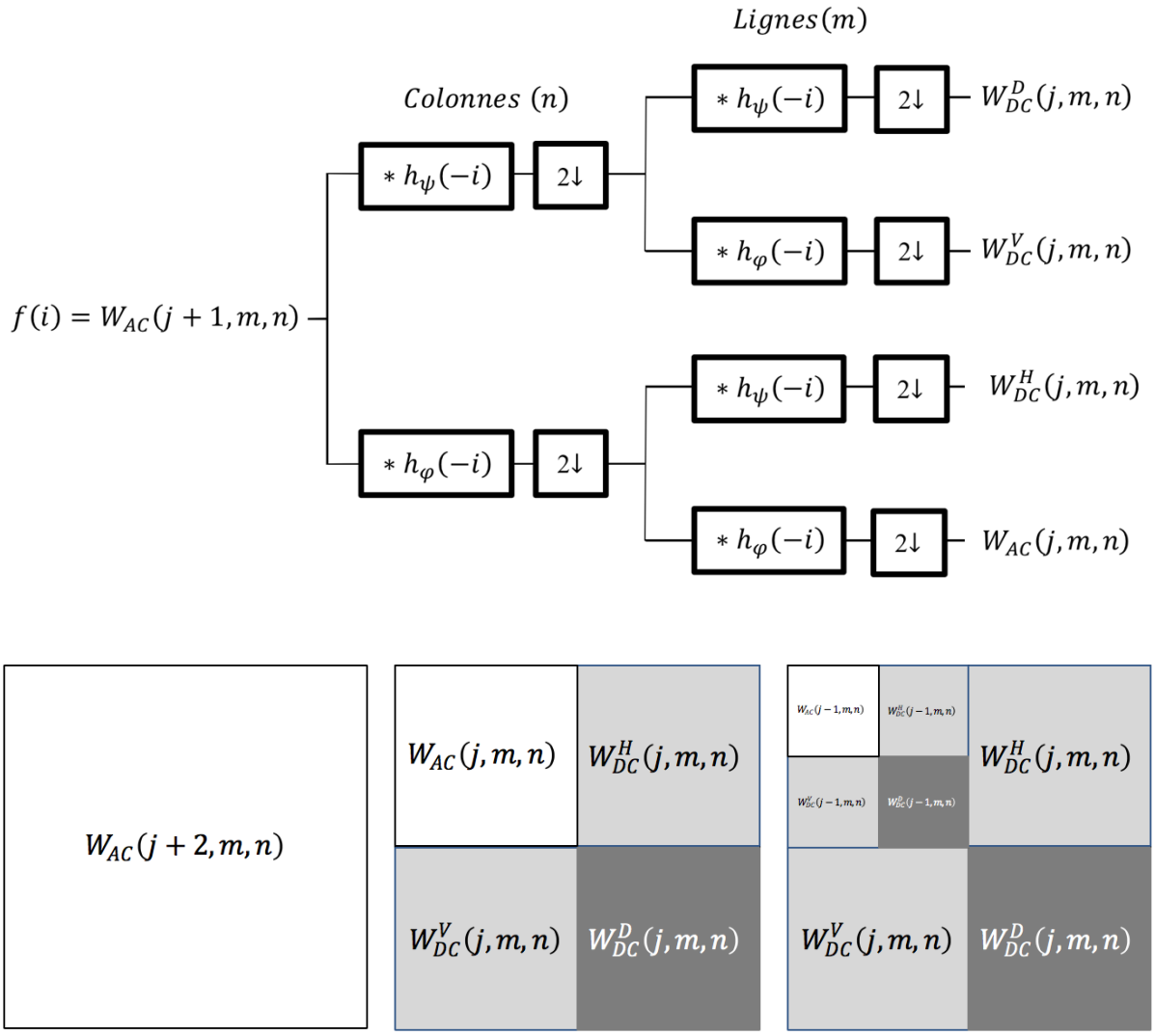
$$W^i_{DC}(j,m,n) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(x,y)\, \psi^i_{j,m,n}(x,y) \tag{4.21}$$

Avec $W_{AC}(j_0,m,n)$ est le coefficient d'approximation de $f(x,y)$ à l'échelle j_0 et $W^i_{DC}(j,m,n)$ sont les coefficients horizontal, vertical et diagonal de détail à l'échelle j où : $j \geq j_0$.

Pour calculer la 2D-IDWT :

$$f(x,y) = \frac{1}{\sqrt{MN}} \left(\sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} W_{AC}(j_0,m,n)\, \varphi_{j_0,m,n}(x,y) + \sum_{i=H,V,D} \sum_{j=j_0} \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} W^i_{DC}(j,m,n)\, \psi^i_{j,m,n}(x,y) \right) \tag{4.22}$$

La transformée en ondelettes à deux dimensions 2D-IDWT présentée par la figure suivante :



- **Application DWT pour les images :**

Le principe de l'algorithme consiste à décomposer le signal en deux pour chaque itération, une approximation aux informations les plus importantes ou les basses fréquences (L en anglais de ‘’low’’), et en ses détails ou composantes hautes fréquences.

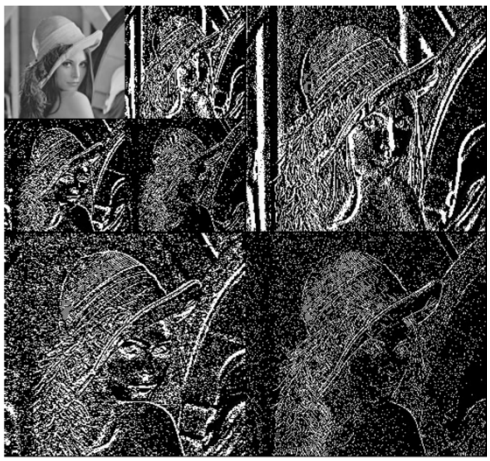
Les mêmes procédures sont réitérées, toujours sur l’approximation uniquement, jusqu’à l’obtention de la résolution souhaitée. Dans le cas d'une image (un signal bidimensionnel), deux étapes sont nécessaires pour obtenir chaque niveau de décomposition. Une pour les lignes et la deuxième pour les colonnes (voir la figure précédente). Donc, pour chaque niveau *i* de décomposition on a une approximation (LL*i* pour low-Low niveau *i*) et trois détails LH*i*, HL*i* et HH*i* :

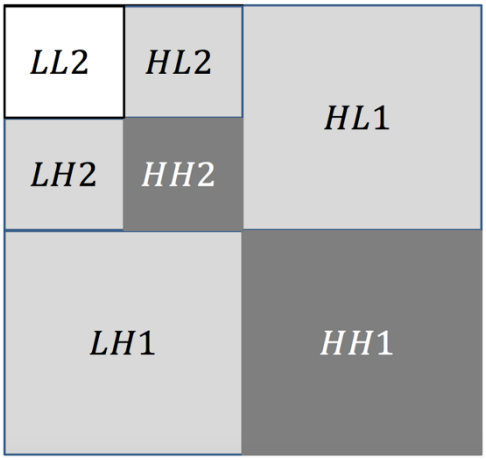
- Low-High niveau i LHi : une décomposition basse fréquence selon lignes suivie par une décomposition haute fréquence pour les colonnes. L’indice *i* pour indiquer sur le *i-ieme* niveau de décomposition.

- High-Low niveau *i* HL*i*: une décomposition haute fréquence selon lignes suivie par une décomposition basse fréquence pour les colonnes. L'indice *i* pour indiquer sur le *i-ieme* niveau de décomposition.
- HighHigh niveau *i* HH1 : une décomposition haute fréquence selon lignes suivie par une décomposition haute fréquence pour les colonnes. L'indice *i* pour indiquer sur le *i-ieme* niveau de décomposition.

Représentation des coefficients d'approximation et de détails

image originale

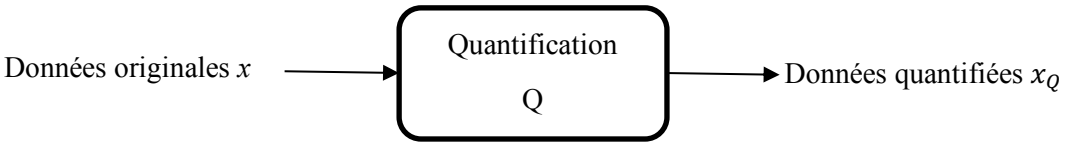




IV.3. Quantification

Cette étape est la principale étape de perte de données. L'objectif de la quantification est de déterminer la valeur la plus proche de dans l'ensemble Q à partir d'une valeur d'entrée donnée dans l'espace F. Lorsqu'un nombre x est quantifié en L niveaux, cela signifie que sa valeur est remplacée (ou quantifiée) par le membre le plus proche d'un ensemble de niveaux de quantification.

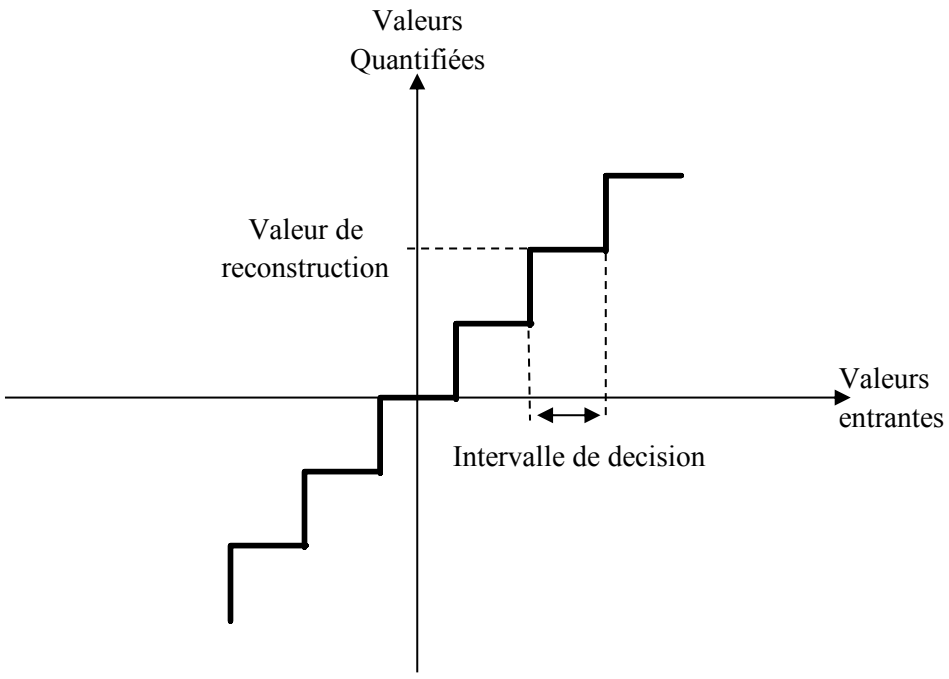
Nous pouvons quantifier directement une image ou quantifier les résultats d'une transformation comme décrite plus haut.



Il existe plusieurs types de quantification tels que la quantification scalaire.

IV.3.1. La quantification scalaire

La quantification scalaire est l'une des idées les plus simples et les plus générales pour la compression avec perte. Où il s'agit d'une approximation de chaque valeur du signal aléatoire $x(t)$, par une valeur qui appartient à un ensemble fini de codes $\{y(1), y(2), \dots, \}$. Pour toute amplitude x comprise dans l'intervalle $[x(l-1), x(l)\ ]$, on fait correspondre une valeur quantifiée située dans cet intervalle.



Attention ! :

- La quantification est une opération non-linéaire : $(x + y)_q \neq x_q + y_q$
- La quantification est une opération irréversible → codage avec pertes

Une quantification scalaire est caractérisée par :

- Nombre de niveaux quantifiés $\mathrm{L} + 1 = 2^b$ (b bits).
- Pas de quantification : $\Delta = x(k) - x(k-1)$
- Erreur de quantification : $e = x - y$
- Erreur de granulation : $|e| \leq \frac{\Delta}{2}$

Il existe plusieurs méthodes de quantificateurs scalaire :

A. Quantification scalaire par échelle : On garde seulement les coefficients correspondants à une échelle de quantification (arrondissement).

Exemple : Soit le signal d'origine $x = [2 \quad 1 \quad 7 \quad 4]$

Signal transformé $F = dct(x)$

$$F = [7.0000 \quad -2.9302 \quad -1.0000 \quad 3.3785]$$

Quantification scalaire sur une échelle absolue de 8 valeurs, [0 1 2 3 4 5 6 7 8]

Signal quantifié $x_q = [7 \quad -3 \quad -1 \quad 3]$

Signal reconstruit (DCT inverse) $idct(x_q)$

$$\hat{x} = [1.8519 \quad 1.2284 \quad 6.7716 \quad 4.1481]$$

B. Quantification scalaire par Seuil : On définit une valeur fixée à priori (Threshold) qui permettra d'annuler toutes les valeurs inferieures à celle-ci.

Exemple : Si on reprend un exemple de la transformée DWT : Soit le signal rampe suivant :

$$x = [2 \quad 1 \quad 11 \quad 11 \quad 17 \quad 3 \quad 0 \quad 1]$$

Les coefficients d'ondelettes avec 1 niveau calculés par Matlab dwt (x,' haar ') :

$$CA = [2.1213 \quad 15.5563 \quad 14.1421 \quad 0.7071]$$

$$CD = [0.7071 \quad 0 \quad 9.8995 \quad -0.7071]$$

Quantification $x_q = 0\ (\leq 0)$ avec arrondissement

$$x_q = [2 \quad 16 \quad 14 \quad 1 \quad 1 \quad 0 \quad 10 \quad 0]$$

Signal reconstruit en Matlab idwt (A,D,' haar ')

$$\hat{x} = [2.1213 \quad 0.7071 \quad 11.3137 \quad 11.3137 \quad 16.9706 \quad 2.8284 \quad 0.7071 \quad 0.7071]$$

C. Quantification scalaire par Nombre : on définit un nombre de valeur à garder parmi l'ensemble des coefficients, Dans ce cas le seuil est défini par rapport au nombre et non pas le niveau de valeur.

Exemple : Toujours avec l'exemple précèdent de la transformée DWT :

$$x = [2 \quad 1 \quad 11 \quad 11 \quad 17 \quad 3 \quad 0 \quad 1]$$

$$CA = [2.1213 \quad 15.5563 \quad 14.1421 \quad 0.7071]$$

$$CD = [0.7071 \quad 0 \quad 9.8995 \quad -0.7071]$$

Les coefficients quantifies seront calculé par rapport au nombre 5/8 (c-à-d on doit garder uniquement 5 coefficients sur 8) :

$$x_q = [2.1213 \quad 15.5563 \quad 14.1421 \quad 0.7071 \quad 0 \quad 0 \quad 9.8995 \quad 0]$$

Signal reconstruit en Matlab idwt (AC,DC,' haar ')

$$\hat{x} = [1.5 \quad 1.5 \quad 11 \quad 11 \quad 17 \quad 3 \quad 0.5 \quad 0.5]$$

D. Quantification scalaire par Matrice : Cette quantification est réalisée par une matrice de quantification Q de la taille de la matrice d'origine, Elle s'applique comme suit :

$$F_Q(k_1, k_2) = \left\lfloor \frac{F(k_1, k_2)}{Q(k_1, k_2)} \right\rfloor \tag{4.23}$$

Ou $\lfloor . \rfloor$ est l'entier directement inférieur. Avec : F_Q est la matrice quantifiée, F est la matrice des coefficients à quantifier et Q est la table de quantification. Dans les traitements d'images actuels, la table de quantification est définie par :

$$Q(k_1, k_2) = 1 + (k_1 + k_2) \times R \tag{4.24}$$

Où R représente le facteur de qualité de la matrice de quantification.

***Exemple* :** Soit un bloc de 8×8 pixels à 256 niveaux de gris :

$$f = \begin{bmatrix} 76 & 76 & 76 & 255 & 255 & 255 & 255 & 255 \\ 76 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 76 & 76 & 149 & 149 & 149 & 255 & 255 & 255 \\ 76 & 255 & 149 & 255 & 149 & 29 & 29 & 29 \\ 76 & 255 & 149 & 255 & 149 & 29 & 255 & 255 \\ 255 & 255 & 149 & 149 & 149 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 29 & 29 \end{bmatrix}$$

Ensuite la Transformée en DCT est appliqué sur les pixels de chaque bloc ($F = dct2(f)$).

$$F = \begin{bmatrix} 1474 & -3 & -38 & -128 & 49 & -8 & -141 & 11 \\ 1 & -171 & -51 & 35 & -45 & -4 & 40 & -58 \\ 93 & -2 & -43 & 45 & 26 & 0 & 38 & 24 \\ 61 & -66 & 70 & -11 & 23 & 15 & -48 & 6 \\ -113 & 62 & -71 & -3 & 27 & -10 & -25 & -33 \\ -47 & 1 & -5 & 18 & 28 & 14 & 4 & 2 \\ -72 & -37 & 37 & 51 & 7 & 18 & 33 & -1 \\ 41 & -100 & 21 & -9 & 2 & 27 & 7 & 7 \end{bmatrix}$$

Les valeurs de la matrice DCT seront divisées par la matrice de quantification Q.

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Ensuite, la matrice DCT quantifiée F_Q est donnée par :

$$F_Q = \begin{bmatrix} 92 & -1 & -4 & -8 & 2 & -1 & -3 & 0 \\ 0 & -15 & -4 & 1 & -2 & -1 & 0 & -2 \\ 6 & -1 & -3 & 1 & 0 & 0 & 0 & 0 \\ 4 & -4 & 3 & -1 & 0 & 0 & -1 & 0 \\ -7 & 2 & -2 & -1 & 0 & -1 & -1 & -1 \\ -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

IV.4. Codage entropique

Concernant partie du codage, la sélection de la méthode de compression avec pertes est selon le type de transformée utilisée, par exemple, pour le format GIF, c'est le codage LZW qui est utilisé, pour le JPEG c'est RLE et Huffman et pour pratiquement toutes les méthodes à base d'ondelettes c'est le codage arithmétique qui est utilisé.

IV.5. Différence entre la compression avec perte et sans perte

La compression avec perte et la compression sans perte sont les deux termes largement catégorisés sous les méthodes de compression de données. La principale différence entre les deux méthodes est que la première produit une correspondance proche des données après la décompression alors que la seconde crée des données d'origine exactes.

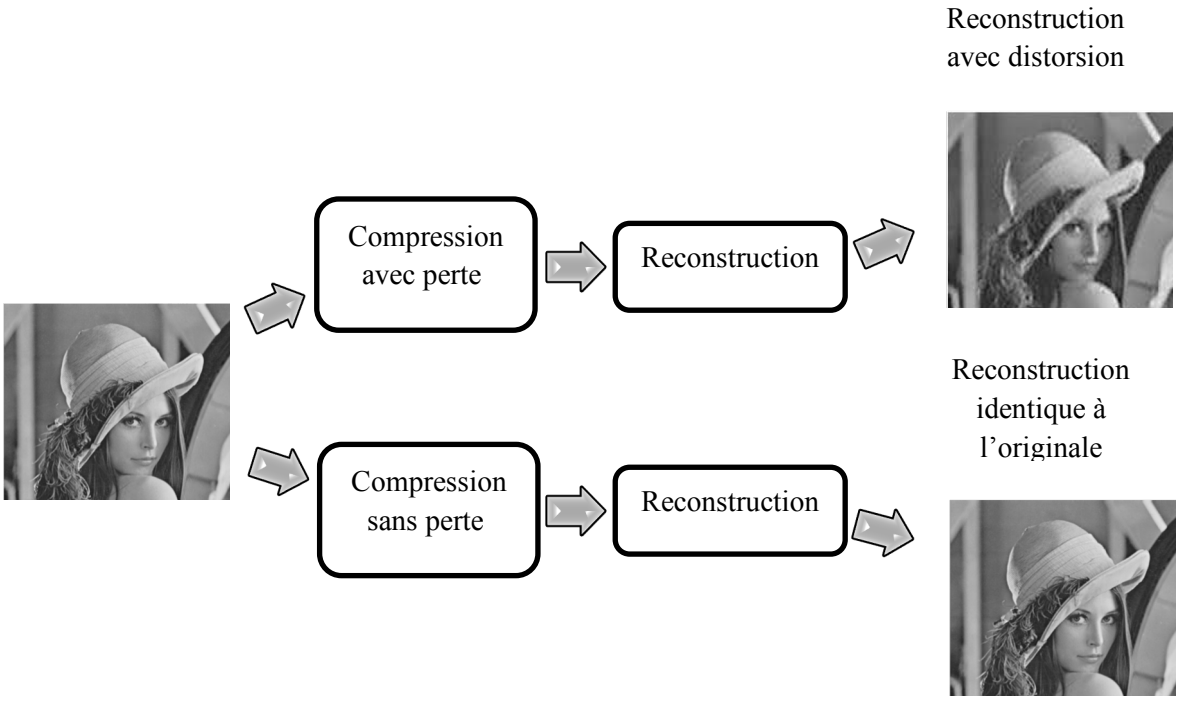


Table de comparaison :

	Compression avec perte	**Compression sans perte**
Définition	La compression avec perte est la méthode de codage qui utilise des estimations imprécises pour représenter le contenu.	La compression sans perte est un groupe d'algorithmes de compression de données qui permet de reconstruire avec précision les données d'origine à partir des données compressées.
Utilisé dans	Images, audio et vidéo.	Texte ou programme, images radiologique, données satellitaires et son.
Exemples	• Par moyennage de blocs • Par transformation linéaire optimale ou de Karhunen-Loeve (KLT) • Par transformée en cosinus discrète (DCT) : JPEG • Par transformée en ondelettes discrète (DWT) : JPEG 2000 • Par quantification (scalaire ou vectorielle)	• Codes à longueur variable : VLC préfixé, Shannon-Fano, Huffman. • Codage arithmétique • Codage à base de dictionnaire : Lempel-Ziv (LZ77, LZW) • Codage par décorrélation : Run-Length Encoding (RLE), prédictif sans perte

	• Par détection de motifs images redondants (fractale)	
Application	JPEG, GUI, MP3, MP4, OGG, H-264, MKV, etc.	RAW, BMP, PNG, WAV, FLAC, ALAC etc.
Capacité de stockage de données	Plus	Moins par rapport à la méthode avec perte
Remarques	• Bon taux de compression • Perte d'information	• Taux de compression limité • Aucune perte d'information

IV.6. Critères d'évaluation

Lorsqu'on utilise des méthodes non réversibles pour la compression d'images, il est utile de pouvoir mesurer la perte de qualité de l'image reconstruite par rapport à l'image originale afin de pouvoir optimiser la compression. L'évaluation de la qualité, ou plutôt la mesure de la dégradation subie par un système ou une opération de traitement tel que la compression, consiste à mesurer la différence entre les valeurs du signal ou d'image originales et reconstruites. Cette différence dépend essentiellement du rapport de compression, qui est défini par :

$$CR = \frac{Nombre\ de\ bits\ d'information\ originale}{Nombre\ de\ bits\ d'information\ reconstruite} \tag{4.25}$$

A partir de ce rapport, le taux de compression en pourcentage (T_c) est donné par l'équation ci-dessous :

$$T_c = \left(1 - \frac{1}{CR}\right) \times 100 \tag{4.26}$$

On trouve beaucoup paramètres ou critères d'évaluation de la qualité utilisés pour mesurer les performances des méthodes de compression sont, par exemple :

IV.6.1. Erreur moyenne absolue (Mean absolute error :MAE)

Cette méthode compare la différence entre la valeur de pixel d'origine et la valeur reconstruite, puis trouve la moyenne pour l'ensemble de l'image :

$$MAE = \frac{1}{N}\sum_{n=1}^{N} |x_n - \hat{x}_n| \tag{4.27}$$

Où N est le nombre total de pixels dans l'image, x_n la valeur du pixel n dans l'image d'origine et $\hat{x}_n$ la valeur du pixel n dans l'image compressée.

IV.6.2. Erreur moyenne quadratique (Mean squared error : MSE)

Cette méthode compare la différence entre la valeur de pixel d'origine et la valeur reconstruite, et la met au carré avant de trouver la valeur moyenne pour toute l'image :

$$MSE = \frac{1}{N}\sum_{n=1}^{N} |x_n - \hat{x}_n|^2 \tag{4.28}$$

IV.6.3. L'indice de la SIMilarité Structurelle (Structural SIMilarity Index : SSIM)

L'indice de la similarité Structurelle (SSIM) est une mesure de similarité entre les structures de deux images reconstruite et originale

$$SSIM = I(x,\hat{x}).C(x,\hat{x}).S(x,\hat{x}) \tag{4.29}$$

I est la luminance, C chrominance et S la structure.

- La comparaison de la luminance, est déterminée par l'expression suivante :

$$I(x,\hat{x}) = \frac{2\mu_x\mu_{\hat{x}} + c_1}{\mu_x^2 + \mu_{\hat{x}}^2 + c_1} \tag{4.30}$$

Où : $\mu_x = \frac{1}{N}\sum_{n=1}^{N} x_n$ et $c_1 = (K_1 L)^2$. $K_1 \ll 1$ et L est le rang dynamique des valeurs des pixels (255 pour une image de niveau de gris codé sur 8 bits).

- La fonction de comparaison de chrominance prend la forme suivante :

$$C(x,\hat{x}) = \frac{2\sigma_x\sigma_{\hat{x}}}{\sigma_x^2 + \sigma_{\hat{x}}^2 + c_2} \tag{4.31}$$

Où : $\sigma_x = \sqrt{\mu_{x^2} - \mu_x^2}$ et $c_2 = (K_2 L)^2$. $K_2 \ll 1$.

- La fonction de comparaison de structure est défini comme suit :

$$S(x,\hat{x}) = \frac{\sigma_{x\hat{x}} + c_3}{\sigma_x\sigma_{\hat{x}} + c_3} = \frac{Cov(x,\hat{x}) + c_3}{\sigma_x\sigma_{\hat{x}} + c_3} \tag{4.32}$$

Où : $Cov(x,\hat{x}) = \mu_{x\hat{x}} - \mu_x\mu_{\hat{x}}$ et $\mu_{x\hat{x}} = \frac{1}{N}\sum_{n=1}^{N} x_n\hat{x}_n$.

IV.6.4. Rapport signal / bruit (Signal-to-noise ratio : SNR)

Le SNR compare la puissance du signal d'origine avec l'erreur quadratique moyenne (MSE) et donnée par :

$$SNR = \frac{P}{MSE} \tag{4.33}$$

Où P est la puissance du signal d'origine ;

$$P = \frac{1}{N}\sum_{n=1}^{N}(x_n)^2 \tag{4.34}$$

On peut le convertir en décibels par :

$$SNR(dB) = 10\log_{10}(SNR) \tag{4.35}$$

IV.6.5. Le pique du rapport signal sur bruit (Peak signal-to-noise ratio : PSNR)

La mesure la plus couramment utilisée par la communauté internationale est la mesure du rapport signal à bruit en pic PSNR (Pic Signal to Noise Ratio). Il est défini par le rapport entre la puissance maximale possible du signal et le bruit en termes de MSE. Il est exprimé en décibels comme suit :

$$PSNR(dB) = 20\log_{10}\left(\frac{max}{\sqrt{MSE}}\right) \tag{4.36}$$

Où " max" est la valeur de pixel maximale possible dans l'image. La faible valeur du PSNR signifie que l'image est de mauvaise qualité.

IV.6. Normes et les organismes de normalisation de compression d'images

IV.7.1 Organismes de Normalisation

Depuis les années 80 du siècle dernier, les méthodes de codages de compression des données (codage/décodage) notamment des images qui sont standardisées par des organismes mondiaux reconnu Parmi ces organismes, on peut citer :

UIT (Union Internationales de Telecom) : subdivise en trois secteurs principaux:

- IUT-D (secteur Développement des télécommunications),
- IUT-R(secteur des Radiocommunications) et
- IUT-T (secteur de la normalisation des Télécommunications).

ISO (International Standard Organisation),
CCITT (Comité Consultatif International Télégraphique et Téléphonique),
CCIR (Comité Consultatif International pour la Radiodiffusion).

IV.7.2 Standards de Compression multimédia

Les formats de la multimédia compressée sont maintenant nombreux et beaucoup sont normalisés. Ci-dessous quelques exemples de standards de compression de multimédia qu'on utilise avec les organismes de standardisation.

- **Standards d'image**

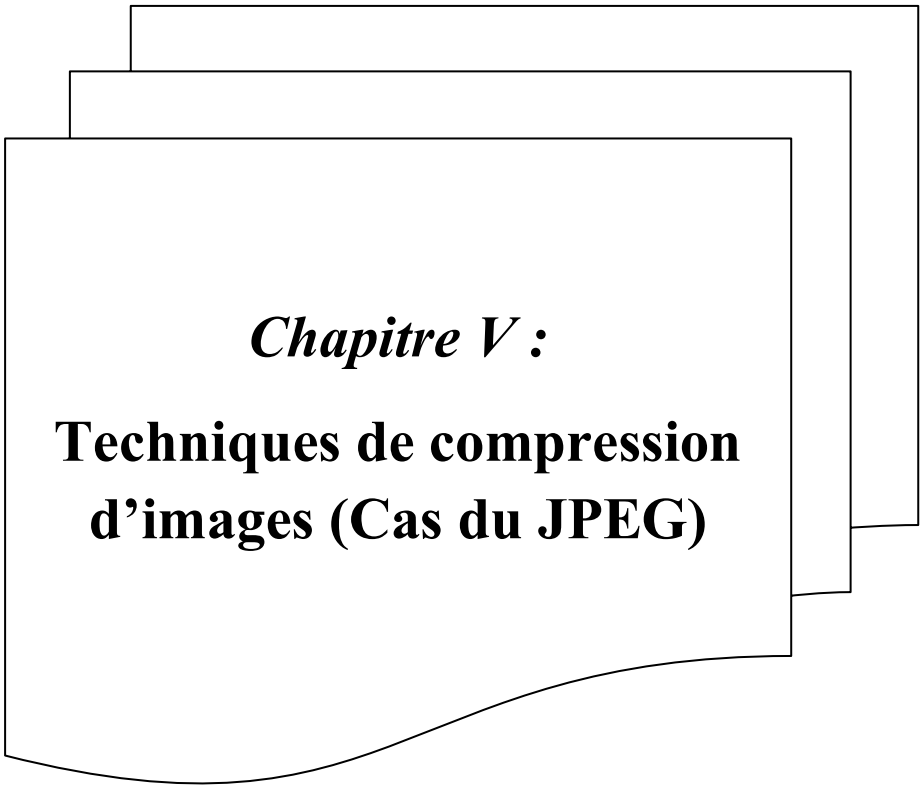
1992 : La norme JPEG (Joint Photographic Expert Group).
2000 : La norme JPEG 2000.

- **Standards du son**

1987-88 : Norme G.711 - IUT-REC-G.711 (compression audio par codages PCM-U et PCM-A).
1991 : Deux formats étaient disponibles (MUSICAM et ASPEC).
1992 : La norme AC-3 (compression numérique pour le son).
1993 : La norme AAC - ISO/CEI 13818-7 (une extension du MPEG-2).
1999 : La norme MP3 - ISO/CEI 11172-3 (ou MPEG-1/2 Audio Layer III) pour la compression du son numérique.

- **Standards Vidéo**

1988 : fondement de MPEG (Moving Picture Coding Experts Group) ;
1993 : La norme MPG1 (qualité VHS sur un CD-ROM)
1994 : La norme MPEG2 (vidéo pour application télévisuelle)
1998 : La norme MPEG4 (TV Haute définition)
2000 : La norme MPEG7 (pour le multimédia)



Chapitre V :

Techniques de compression d'images (Cas du JPEG)

V. Techniques de compression d'images (Cas du JPEG)

V.1. Généralités

Les méthodes de compression d'images sont standardisées et suivent donc un processus connu et clairement défini. Parmi ces standards, le Joint Photographic Expert Group (JPEG) est la première norme internationale de la compression d'images pour les images fixes en couleur et en niveaux de gris.

JPEG ou ISO/CEI 10918-1 UIT-T Recommendation T.81, est le résultat du développement des travaux entamés entre 1978 et 1980, Spécifiée en 1991 et créée en 1992. Est une norme de compression d'images commune par l'Union de Télécommunication Internationale (ITU), l'Organisation internationale de normalisation (ISO) et Commission électrotechnique internationale (IEC).

V.2. Principe de la compression JPEG

La compression d'une image au format JPEG suit un certain nombre d'étapes pour réduire l'espace occupé par différentes méthodes. La figure suivante illustre ces étapes :

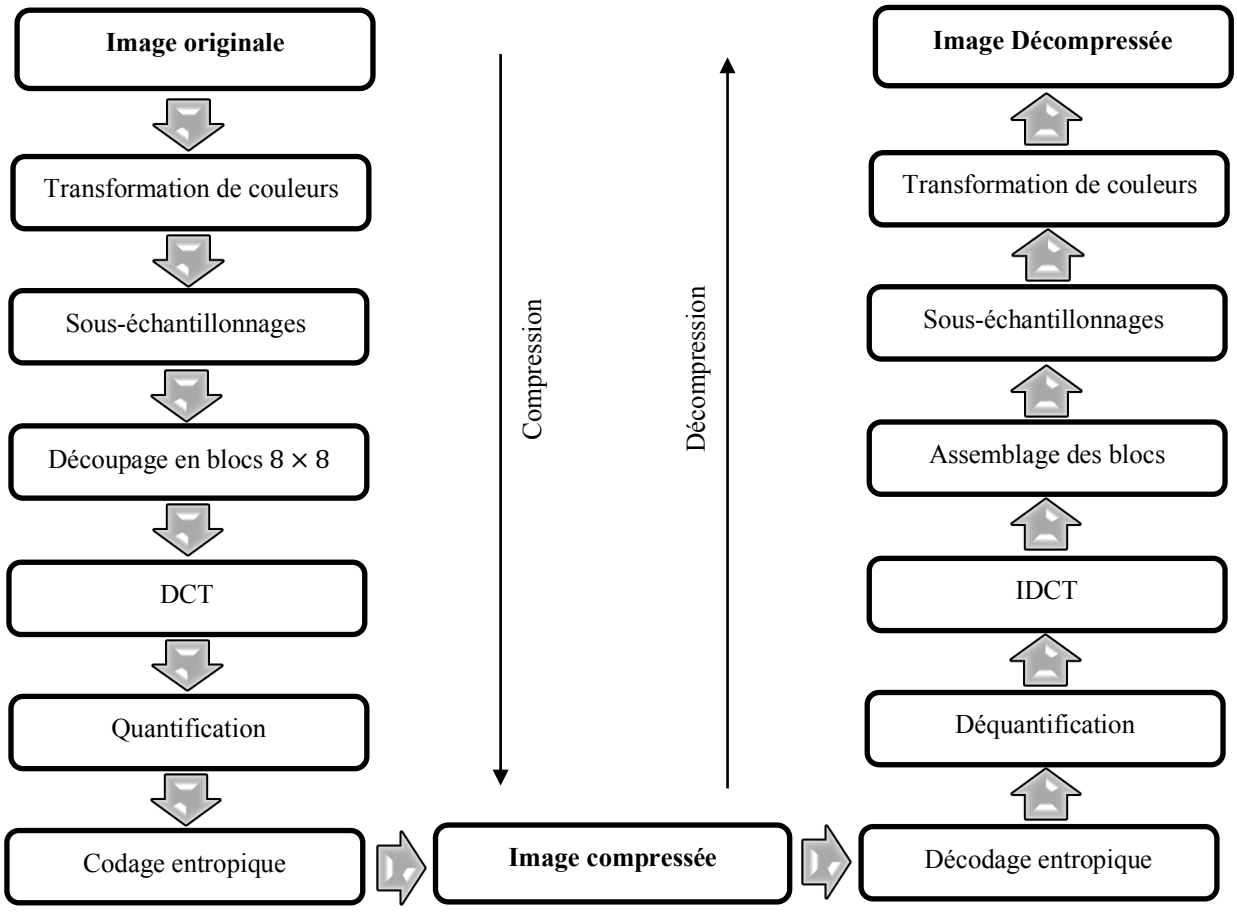


Figure V.1 : Schéma bloc de la compression/décompression JPEG.

1. Transformer l'image en un espace de couleurs optimale.
2. Sous-échantillonnage de la chrominance.
3. Découpage de chaque plan en blocs de 8 × 8 pixels.
4. Appliquer la DCT à chaque bloc de pixels pour retirer la redondance des données de l'image.
5. Quantifier chaque bloc de coefficients DCT avec des fonctions optimisées pour l'œil humain.
6. Ordonner les coefficients suivant un balayage en Zig-Zag pour placer d'abord les coefficients correspondant aux fréquences les plus basses.
7. Coder le résultat en utilisant un codage entropique pour retirer les redondances résiduelles.

Maintenant, nous allons décrire toutes ces étapes qui permettent la compression. Puisque la décompression implique les mêmes étapes dans l'ordre inverse, nous ne les répéterons pas.

V.3. Transformation RGB vers YC_bC_r

Dans une image non comprimée, chaque pixel est habituellement caractérisé par 3 composantes de base dans l'espace colorimétrique RGB (Red Green Blue). Dans cette étape, on va passer d'un espace colorimétrique du type RGB à un autre de type de luminance/chrominance YC_bC_r (Cette étape est sautée pour des images aux niveaux de gris).

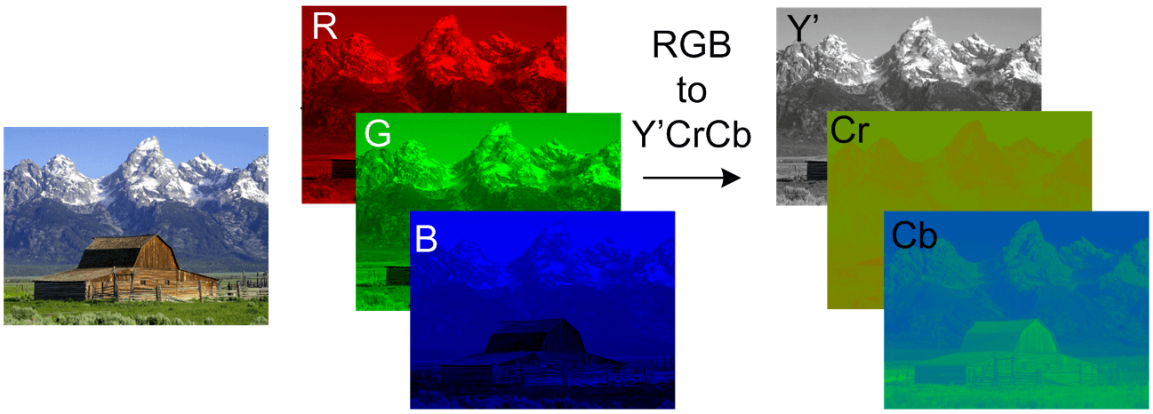


Figure V.2 : Exemple d'un passage du modèle colorimétrique RGB vers YC_bC_r

Le modèle YC_bC_r définit le pixel par trois paramètres, une pour l'information de la luminance (intensité de la lumière) et deux pour la chrominance (la couleur du pixel).

L'avantage d'utiliser ce type de représentation est que :

- L'œil humain est moins sensible aux détails de l'information de couleur (chrominance) que de ceux de l'intensité (luminance).
- Possibilité de compresser davantage la chrominance sans perte visible de qualité (JPEG);
- Possibilité de réserver une bande passante plus étroite (1/4) à la chrominance (signal TV).

La transformation du RGB à YC_bC_r est faite par l'expression mathématique suivante :

$$\begin{cases} Y = \omega_R R + (1 - \omega_B - \omega_R)G + \omega_B B \\ C_b = \dfrac{0.5}{1 - \omega_B}(B - Y) \\ C_r = \dfrac{0.5}{1 - \omega_R}(R - Y) \end{cases} \tag{5.1}$$

$$\begin{cases} Y = \omega_R R + (1 - \omega_B - \omega_R)G + \omega_B B \\ C_b = \dfrac{0.5}{1 - \omega_B}(-\omega_R R - (1 - \omega_B - \omega_R)G + (1 - \omega_B)B) \\ C_r = \dfrac{0.5}{1 - \omega_R}\big((1 - \omega_R)R - (1 - \omega_B - \omega_R)G - \omega_B B\big) \end{cases}$$

Ou ITU : $\omega_R = 0.299$ et $\omega_B = 0.114$.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} \omega_R & 1 - \omega_B - \omega_R & \omega_B \\ -\dfrac{0.5\omega_R}{1 - \omega_B} & -\dfrac{0.5(1 - \omega_B - \omega_R)}{1 - \omega_B} & 0.5 \\ 0.5 & -\dfrac{0.5(1 - \omega_B - \omega_R)}{1 - \omega_R} & \dfrac{0.5\omega_B}{1 - \omega_R} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{5.2}$$

La valeur $Y = 0.299\,R + 0.587\,G + 0.114\,B$ de la luminance.

Les valeurs C_b et C_r s'appellent les valeurs de chrominance. Afin de représenter ces valeurs dans des nombres entiers de 8 bits, ils sont décalés, donc suivant cette saturation l'ajout de 128 à chaque échantillon permet d'obtenir des valeurs entre 0 et 255.

Par conséquent, la transformation ci-dessus peut être exprimée comme :

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{5.3}$$

La transformation inverse de YC_bC_r vers RGB est faite par :

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344 & 0.714 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

V.4. Le sous-échantillonnage

Le but ici est de réduire les informations occupées par la chrominance. Si l'œil humain est très sensible à la luminance, elle ne perçoit pas bien la dégradation dur la chrominance. La perte d'informations sur ce point reste donc relativement inaperçue.

On distingue 3 types majeurs de sous-échantillonnages :

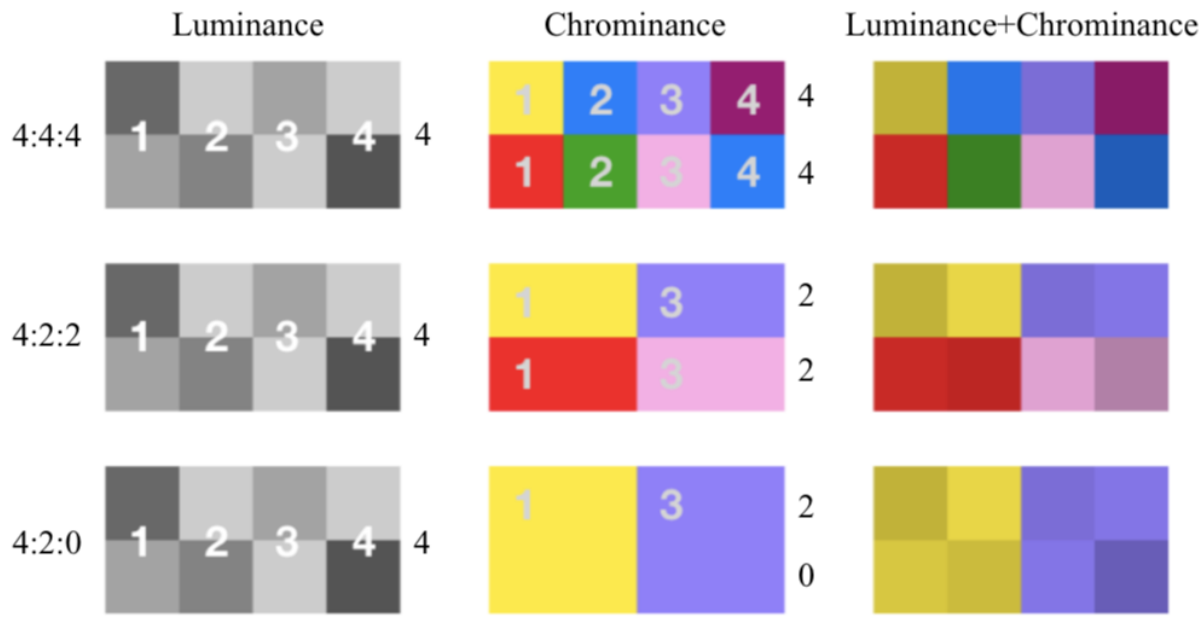


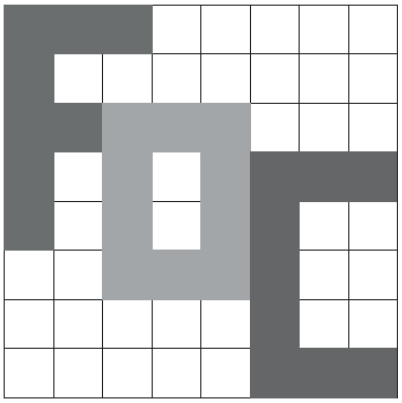
Figure V.3 : Les 3 types de sous-échantillonnages

- Le 4:4:4 donne une augmentation notable de la qualité, puisque qu'il consiste à ne rien faire (sans sous-échantillonnage). Donc aucune compression n'est effectuée et aucune perte de qualité ne peut être enregistrée.
- Le 4:2:2 produit une image de bonne qualité mais moins détaillée, puisque qu'il consiste à supprimer les informations d'un colonne sur deux. Ensuite, la colonne restante contient la moyenne des deux colonnes. Pour la décompression, il suffira de recopier cette information sur les deux colonnes concernées.
- Le 4:2:0 est un sous-échantillonnage qui consiste a effectué les même procédures que le 4:2:2 mais en effectuant aussi cette opération sur les lignes. On divise donc ici par 4 l'information de la chrominance.

V.5. Le découpage en bloc de pixels

À partir de cette étape, la compression sera appliquée aux blocs de pixels seulement. Ces blocs sous formes des matrices 8×8 contiennent 64 pixels, Si la largeur et la longueur de l'image ne sont pas des

multiples de 8, on rajoute des colonnes en répétant la dernière jusqu'à atteindre un multiple de 8 (même traitement pour les lignes). Chaque bloc sera traité individuellement.



$$f = \begin{bmatrix} 76 & 76 & 76 & 255 & 255 & 255 & 255 & 255 \\ 76 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 76 & 76 & 149 & 149 & 149 & 255 & 255 & 255 \\ 76 & 255 & 149 & 255 & 149 & 29 & 29 & 29 \\ 76 & 255 & 149 & 255 & 149 & 29 & 255 & 255 \\ 255 & 255 & 149 & 149 & 149 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 29 & 29 \end{bmatrix}$$

Figure V.4 : Un bloc de 8 × 8 d'une image simple.

Les raisons de ce découpage sont :

- L'application de la DCT à tous 8 × 8 pixels d'une image produit une meilleure compression mais comporte également beaucoup d'opérations arithmétiques et donc lente ; et l'application de la DCT aux unités de données réduit le taux de compression en revanche elle est plus rapide.
- L'expérience montre que, dans une image, les corrélations entre les pixels sont de gamme courte. Un pixel dans une telle image a une valeur (composant de couleur au niveau de gris) qui est proche de ses voisins, mais n'a rien avoir avec les valeurs des pixels lointains.

V.6. Application de la 2D-DCT

Dans le chapitre précédent, nous avons décrit le concept général de la transformée DCT. Cette transformation est effectuée pour connaître les fréquences d'une information en 2 dimensions k_1, k_2.

La norme JPEG applique la DCT pas à l'image entière mais aux unités de données (blocs) de 8 × 8 pixels.

$$F(k_1, k_2) = \frac{1}{4}\alpha(k_1)\alpha(k_2)\sum_{n_1=0}^{7}\sum_{n_2=0}^{7} f(n_1, n_2)\cos\left(\frac{\pi(2n_1+1)k_1}{16}\right)\cos\left(\frac{\pi(2n_2+1)k_2}{16}\right) \quad (5.4)$$

$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{2}} & pour\ \ k = 0 \\ 1 & pour\ \ k \neq 0 \end{cases}$$

La 2D-DCT peut être donnée sous forme matricielle par la formule suivante :

$$C_8 = \frac{1}{2}.\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{1}{16}\pi & \cos\frac{3}{16}\pi & \cos\frac{5}{16}\pi & \cos\frac{7}{16}\pi & \cos\frac{9}{16}\pi & \cos\frac{11}{16}\pi & \cos\frac{13}{16}\pi & \cos\frac{15}{16}\pi \\ \cos\frac{2}{16}\pi & \cos\frac{6}{16}\pi & \cos\frac{10}{16}\pi & \cos\frac{14}{16}\pi & \cos\frac{18}{16}\pi & \cos\frac{22}{16}\pi & \cos\frac{26}{16}\pi & \cos\frac{30}{16}\pi \\ \cos\frac{3}{16}\pi & \cos\frac{9}{16}\pi & \cos\frac{15}{16}\pi & \cos\frac{21}{16}\pi & \cos\frac{27}{16}\pi & \cos\frac{33}{16}\pi & \cos\frac{39}{16}\pi & \cos\frac{45}{16}\pi \\ \cos\frac{4}{16}\pi & \cos\frac{12}{16}\pi & \cos\frac{20}{16}\pi & \cos\frac{28}{16}\pi & \cos\frac{36}{16}\pi & \cos\frac{44}{16}\pi & \cos\frac{52}{16}\pi & \cos\frac{60}{16}\pi \\ \cos\frac{5}{16}\pi & \cos\frac{15}{16}\pi & \cos\frac{25}{16}\pi & \cos\frac{35}{16}\pi & \cos\frac{45}{16}\pi & \cos\frac{55}{16}\pi & \cos\frac{65}{16}\pi & \cos\frac{75}{16}\pi \\ \cos\frac{6}{16}\pi & \cos\frac{18}{16}\pi & \cos\frac{30}{16}\pi & \cos\frac{42}{16}\pi & \cos\frac{54}{16}\pi & \cos\frac{66}{16}\pi & \cos\frac{78}{16}\pi & \cos\frac{90}{16}\pi \\ \cos\frac{7}{16}\pi & \cos\frac{21}{16}\pi & \cos\frac{35}{16}\pi & \cos\frac{49}{16}\pi & \cos\frac{63}{16}\pi & \cos\frac{77}{16}\pi & \cos\frac{91}{16}\pi & \cos\frac{105}{16}\pi \end{bmatrix}$$

Où la matrice transformée en 2D-DCT est donnée par : $F = C_8\ f\ C_8^t$

On obtient une représentation spectrale 8×8 (fréquence horizontale × fréquence verticale).

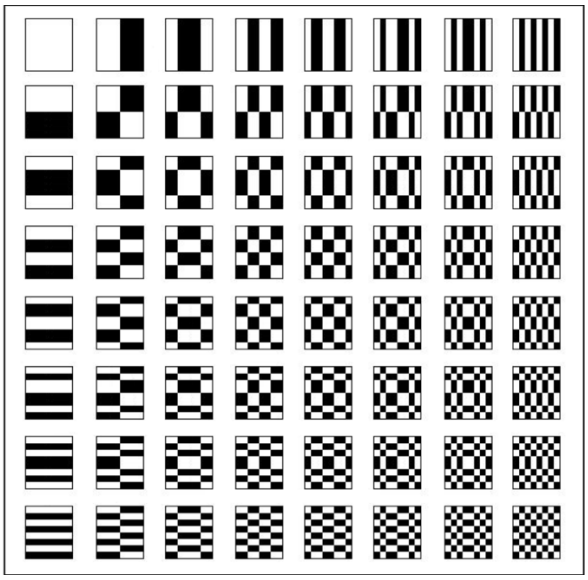


Figure V.5 : La représentation spectrale de la matrice DCT 8×8

Le schéma indique ce que sont :

- Fréquences basses : Peu de variations (bloc en haut à gauche)
- Fréquences verticales croissantes (descendre une colonne)

- Fréquences horizontales croissantes (ligne de gauche à droite)
- Fréquences hautes : Variations rapides (bloc en bas à droite)
- Le point (0,0) d'un bloc est la valeur moyenne des pixels du bloc (fréquence 0).

Pour l'exemple précèdent (Figure V.5) :

$$f = \begin{bmatrix} 76 & 76 & 76 & 255 & 255 & 255 & 255 & 255 \\ 76 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 76 & 76 & 149 & 149 & 149 & 255 & 255 & 255 \\ 76 & 255 & 149 & 255 & 149 & 29 & 29 & 29 \\ 76 & 255 & 149 & 255 & 149 & 29 & 255 & 255 \\ 255 & 255 & 149 & 149 & 149 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 29 & 29 & 29 \end{bmatrix}$$

Transformation en DCT

$$F = \begin{bmatrix} 1474 & -3 & -38 & -128 & 49 & -8 & -141 & 11 \\ 1 & -171 & -51 & 35 & -45 & -4 & 40 & -58 \\ 93 & -2 & -43 & 45 & 26 & 0 & 38 & 24 \\ 61 & -66 & 70 & -11 & 23 & 15 & -48 & 6 \\ -113 & 62 & -71 & -3 & 27 & -10 & -25 & -33 \\ -47 & 1 & -5 & 18 & 28 & 14 & 4 & 2 \\ -72 & -37 & 37 & 51 & 7 & 18 & 33 & -1 \\ 41 & -100 & 21 & -9 & 2 & 27 & 7 & 7 \end{bmatrix}$$

Cette transformation va permettre de séparer les hautes fréquences des basses fréquences, on aura 64 coefficients, le premier coefficient de la matrice représente le coefficient continue DC ou le fondamental du sous bloc de l'image, c'est le coefficient de la fréquence nulle dont l'amplitude est la plus élevée et le reste des coefficients ce sont des coefficients AC décroissant si on tend vers les hautes fréquences. Le fait de séparer les fréquences hautes et les fréquences basses va ainsi permettre la dé-corrélation des pixels de chaque sous bloc de l'image et ainsi réduire la redondance inter pixel, sachant que la transformation seul sans quantification et codage n'est pas une compression, ce n'est juste qu'une étape post-compression.

V.7. Quantification

Pour chaque bloc de 8×8 la quantification consiste à diviser la matrice obtenue après avoir appliquer la DCT, par une autre, appelée matrice de quantification, et qui contient 8×8 coefficients. Cette division atténue les hautes fréquences car l'œil humain est plus sensible aux basses fréquences. Les coefficients leur correspondant sont souvent ramenés à 0.

La situation picturale la plus fréquente correspond à une matrice transformée de la forme :

$$Q = \begin{bmatrix} A & B & B & C & C & D & D & D \\ B & B & C & C & D & D & D & D \\ B & C & C & D & D & D & D & D \\ C & C & D & D & D & D & D & D \\ C & D & D & D & D & D & D & D \\ D & D & D & D & D & D & D & D \\ D & D & D & D & D & D & D & D \\ D & D & D & D & D & D & D & D \end{bmatrix}$$

Où A, B, C, D désignent, en ordre d´décroissant, le poids numérique de la position. Les matrices de quantifications intéressantes sont celles permettant de ‘’choisir’’ la perte de qualité acceptable. Habituellement, on prend pour matrice de quantification une expression simple comme $Q(k_1, k_2)$ pour garantir que les éléments des matrices de quantification commencent petit au coin gauche supérieure et deviennent plus grand au coin inférieur droit.

$$Q(k_1, k_2) = 1 + (k_1 + k_2) \times R \tag{5.5}$$

$Q(k_1, k_2)$: Valeur du pas de quantification dans un bloc, $k_1$et $k_2 = 0,1,2,\ldots, N-1$ et R : Facteur de qualité de la quantification.

Les matrices suivantes montrent les valeurs par défaut de $Q(k_1, k_2)$ obtenues à partir d'études psychophysiques dans le but de maximiser le taux de compression tout en minimisant les pertes de perception dans les images JPEG.

Pour la luminance

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Pour la chrominance

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Si les éléments des matrices de quantification de la luminance et de la chrominance sont représentés par $Q(k_1, k_2)$, un coefficient DCT quantifié à la position (k_1, k_2) est donné par :

$$F_Q(k_1, k_2) = \left\lfloor \frac{F(k_1, k_2)}{Q(k_1, k_2)} \right\rfloor \tag{5.6}$$

Où $F_Q(k_1, k_2)$ est la valeur du coefficient transformé après la quantification, ou $\lfloor . \rfloor$ est l’entier directement inférieur.

Au décodeur, l'étape de la dé-quantification est réalisée comme suit :

$$\hat{F}(k_1, k_2) = F_Q(k_1, k_2).\, Q(k_1, k_2)$$

Où $\hat{F}(k_1, k_2)$ est le coefficient dé-quantifié.

Exemple : Voici un exemple complet de compression et de décompression JPEG, soit F la matrice transformée et Q une matrice de quantification

$$F = \begin{bmatrix} 1474 & -3 & -38 & -128 & 49 & -8 & -141 & 11 \\ 1 & -171 & -51 & 35 & -45 & -4 & 40 & -58 \\ 93 & -2 & -43 & 45 & 26 & 0 & 38 & 24 \\ 61 & -66 & 70 & -11 & 23 & 15 & -48 & 6 \\ -113 & 62 & -71 & -3 & 27 & -10 & -25 & -33 \\ -47 & 1 & -5 & 18 & 28 & 14 & 4 & 2 \\ -72 & -37 & 37 & 51 & 7 & 18 & 33 & -1 \\ 41 & -100 & 21 & -9 & 2 & 27 & 7 & 7 \end{bmatrix}$$

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

La matrice quantifié F_Q est donnée comme :

$$F_Q = \begin{bmatrix} 92 & -1 & -4 & -8 & 2 & -1 & -3 & 0 \\ 0 & -15 & -4 & 1 & -2 & -1 & 0 & -2 \\ 6 & -1 & -3 & 1 & 0 & 0 & 0 & 0 \\ 4 & -4 & 3 & -1 & 0 & 0 & -1 & 0 \\ -7 & 2 & -2 & -1 & 0 & -1 & -1 & -1 \\ -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Exemple : Dans cette section, nous donnons un exemple de codage d'un bloc de 8×8 sélectionné à partir de l'image lena.bmp' de taille 256×256.

$$\begin{bmatrix} 138 & 143 & 145 & 144 & 144 & 142 & 143 & 141 \\ 141 & 144 & 143 & 144 & 145 & 142 & 143 & 137 \\ 141 & 144 & 140 & 144 & 142 & 140 & 141 & 137 \\ 145 & 147 & 143 & 139 & 142 & 141 & 140 & 138 \\ 143 & 143 & 145 & 143 & 139 & 138 & 138 & 131 \\ 143 & 145 & 143 & 144 & 137 & 135 & 135 & 135 \\ 148 & 145 & 144 & 139 & 134 & 134 & 132 & 134 \\ 146 & 144 & 142 & 137 & 134 & 133 & 129 & 134 \end{bmatrix}$$

(a) Bloc original

$$\begin{bmatrix} 235.6250 & -1.0333 & -12.0809 & -5.2029 & 2.1250 & -1.6724 & -2.7080 & 1.3238 \\ -22.5904 & -17.4842 & -6.2405 & -3.1574 & -2.8557 & -0.0695 & 0.4342 & -1.1856 \\ -10.9493 & -9.2624 & -1.5758 & 1.5301 & 0.2029 & -0.9419 & -0.5669 & -0.0629 \\ -7.0816 & -1.9072 & 0.2248 & 1.4539 & 0.8963 & -0.0799 & -0.0423 & 0.3315 \\ -0.6250 & -0.8381 & 1.4699 & 1.5563 & -0.1250 & -0.6610 & 0.6088 & 1.2752 \\ 1.7541 & -0.2029 & 1.6205 & -0.3424 & -0.7755 & 1.4759 & 1.0410 & -0.9930 \\ -1.2825 & -0.3600 & -0.3169 & -1.4601 & -0.4900 & 1.7348 & 1.0758 & -0.7613 \\ -2.5999 & 1.5519 & -3.7628 & -1.8448 & 1.8716 & 1.2139 & -0.5679 & -0.4456 \end{bmatrix}$$

(b) Les coefficients DCT

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

(c) Table de quantification

$$\begin{bmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(d) Coefficients quantifiés et arrondis

$$\begin{bmatrix} 240 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ -24 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(e) Coefficients déquantifiés

$$\begin{bmatrix} 142 & 144 & 147 & 150 & 152 & 153 & 154 & 154 \\ 149 & 150 & 153 & 155 & 156 & 157 & 156 & 156 \\ 157 & 158 & 159 & 161 & 161 & 160 & 159 & 158 \\ 162 & 162 & 163 & 163 & 162 & 160 & 158 & 157 \\ 162 & 162 & 162 & 162 & 161 & 158 & 156 & 154 \\ 160 & 161 & 161 & 161 & 160 & 158 & 156 & 154 \\ 160 & 160 & 161 & 162 & 161 & 160 & 158 & 157 \\ 160 & 161 & 163 & 164 & 164 & 163 & 161 & 160 \end{bmatrix}$$

(f) Bloc restitué

V.8. Balayage en Zig-Zag

A cette étape, la matrice 8×8, F_Q obtenue après avoir appliqué la transformée DCT et la quantification est traversé en Zig-Zag pour construire un vecteur de 64 coefficients (0…63).

La première valeur dans le vecteur (a index 0) correspond à la plus basse fréquence spatiale de l'image. Elle s'appelle DC courant continu ("direct current" en anglais). En incrémentant l'index dans le vecteur, nous obtenons des valeurs correspondantes `a des fréquences plus élevées. Ces 63 coefficients s'appellent AC (courant alternatif) qui sont - en général sensiblement plus petits en module que le coefficient dominant DC. Le schéma de parcours Zig-Zag est :

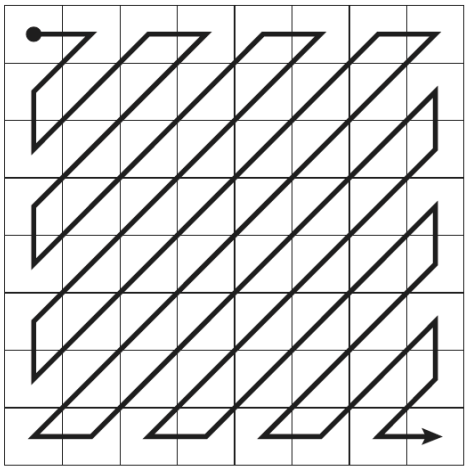


Figure V.6 : Le parcours du bloc 8x8 en Zig-Zag

Selon le parcours en Zig-Zag : les 64 valeurs de l'exemple précèdent F_Q sont écrites comme suit : 92,-1,0,6,-15,-4,-8,-4,-1,4 ……,0.

V.9. Codage entropique

Nous finissons alors par le codage entropique, Cette étape permet d'obtenir une compression supplémentaire sans perte en codant les coefficients DCT quantifiés.

Rappeler : Un codage entropique consiste à coder les symboles qui ont une probabilité d'apparition forte avec de mots de longueur plus faible que les symboles qui ont une probabilité d'apparition faible.

Le coefficient DC et les 63 coefficients AC sont codés séparément. Pour les N_b blocs, le coefficient DC^i de chaque bloc i, qui concentre la majeure partie de l'énergie, varie peu d'un bloc à l'autre.

Ces coefficients $\left(DC^i\right)_{i\in[0,N_b]}$ sont codés en DPCM (Differential Pulse Code Modulation) en utilisant la valeur de coefficient DC de bloc précédent :

DC^0	$DC^1 - DC^0$	$DC^2 - DC^1$	……	$DC^{N_b} - DC^{N_b-1}$

Le but de ce traitement est d'exploiter la corrélation entre les valeurs DC des blocs adjacents et les coder plus efficacement, car ils contiennent typiquement la plus grande partie d'énergie d'image.

Les 63 coefficients AC commençant à partir du coefficient AC_0^i sont codés par plage après le balayage de Zig-Zag.

$$\begin{bmatrix} 92 & -1 & -4 & -8 & 2 & -1 & -3 & 0 \\ 0 & -15 & -4 & 1 & -2 & -1 & 0 & -2 \\ 6 & -1 & -3 & 1 & 0 & 0 & 0 & 0 \\ 4 & -4 & 3 & -1 & 0 & 0 & -1 & 0 \\ -7 & 2 & -2 & -1 & 0 & -1 & -1 & -1 \\ -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zig-Zag sur AC

$$\begin{matrix} -1 & 0 & & & & & & \\ 6 & -15 & -4 & & & & & \\ -8 & -4 & -1 & 4 & & & & \\ -7 & -4 & -3 & 1 & 2 & & & \\ -1 & -2 & 1 & 3 & 2 & -2 & & \\ -2 & 0 & -2 & -1 & 0 & -1 & -3 & \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & -2 & \\ 0 & -1 & -1 & 0 & 0 & 0 & & \\ -1 & 0 & 0 & -1 & 0 & & & \\ -1 & 0 & 0 & 0 & & & & \\ 0 & 0 & 0 & & & & & \\ -1 & 0 & & & & & & \\ 0 & & & & & & & \end{matrix}$$

La séquence $\left(AC_k^i\right)_{k\in[0,63],\ i\in[0,N_b]}$ est ensuite codée bloc par bloc par codage RLE.

Après l'encodage DPCM et RLE, on code les valeurs obtenues à l'aide d'un code de longueur variable : Huffman.

V.10. Avantages et Inconvénients de la compression JPEG

Avantages :

- Standard reconnu par tous, le format JPEG peut être lu par tous les programmes, ordinateurs, Téléphone, tablettes, etc.

- Permet l'obtention de fichiers légers ce qui prend évidemment très peu de place sur la carte mémoire ou le disque dur.
- La perte d'information dû à la compression porte sur des éléments peu sensibles à la vision humaine

Inconvénients :

- Beaucoup de paramètres très techniques à régler (taille des blocs, sous-échantillonnage, matrice de quantification).
- L'algorithme de compression provoque une perte d'information et peut donc entraîner une perte de qualité.
- Pas adapté aux photos de texte, images comportant peu de couleurs ou figures géométriques (pour lesquelles le format GIF est plus adapté).

V.11. Généralités sur la compression JPEG2000

JPEG 2000 abrégé JP2 ou J2K, repose sur un principe de compression totalement différent de JPEG. Son vrai nom est ISO/CEI 15444-1, cette norme est un résultat des efforts de collaboration par l'ISO, la CEI et l'UIT-T, Spécifiée en 1997 et créée en 2000 par le groupe de travail Joint Photographic Experts Group. Depuis mai 2015, il est officiellement reconnu par l'ISO / CEI et l'UIT-T sous le code ISO/IEC CD 154441.

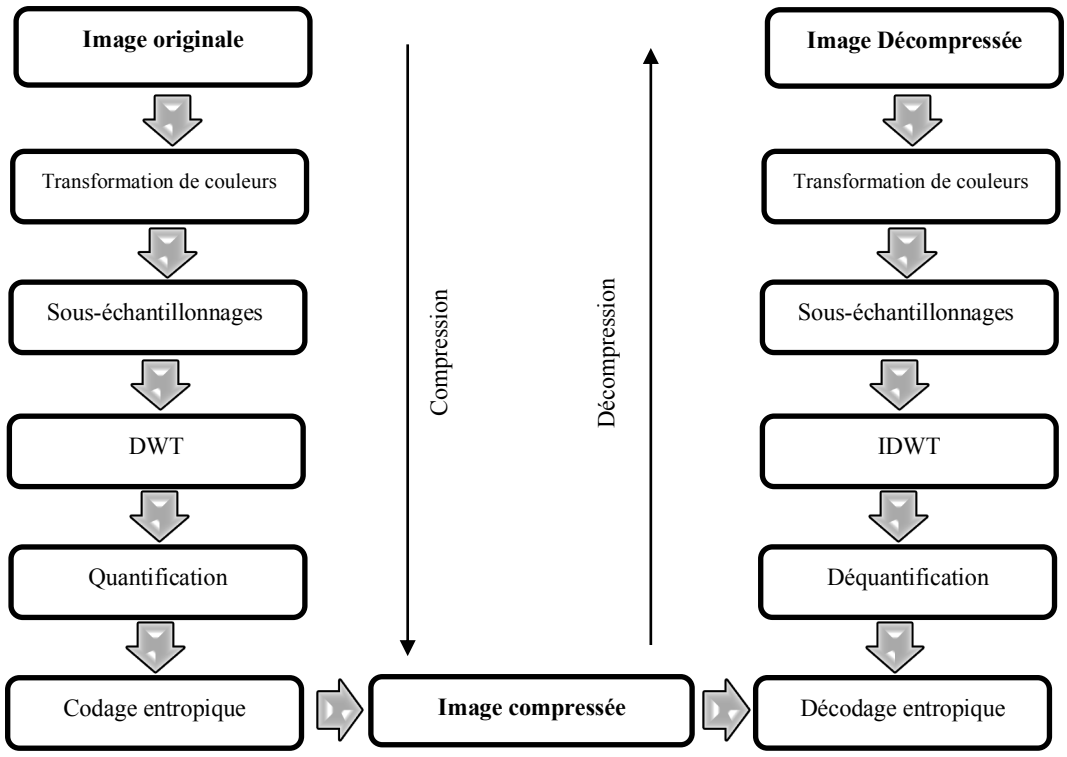


Figure V.7 : Schéma bloc de la compression/décompression JPEG 2000.

Généralement, le schéma synoptique de la compression/décompression JPEG 2000, figure V.7 est fondé sur le même principe que celui de la figure V.1:

- ***Transformation de couleurs*** : Contrairement à la compression JPEG, cette étape n'est pas nécessaire.
- ***Sous-échantillonnage*** : est la même que pour la norme JPEG.
- ***DWT*** : Contrairement à la compression JPEG, le JPEG 2000 basé sur la transformation en ondelettes des pixels. Cette opération agit sur l'image complète et non pas sur des blocs de 8 × 8 pixels. La DWT reproduit plusieurs sous-images par divisions itérative de l'image originale selon un intervalle de fréquences. Généralement, les fréquences hautes sont moins nombreuses que les fréquences basses car les fréquences hautes signifient que les pixels de l'image sont très différents les uns des autres, un phénomène rare dans une image.
- ***La quantification*** **:** C'est l'étape de la destruction, ou les fréquences les plus hautes sont supprimées selon un taux de compression donné.
- ***Codage entropique*** : le JPEG 2000 applique le codage arithmétique adaptatif aux données générées par l'opération de la quantification.

Avantages et des inconvénients du JPEG 2000

Avantages :

- Bien adapté pour la production vidéo et le contenu télévisé en direct.
- Présenté comme étant successeur de JPEG.
- La quantité d'information pertinente préservée est supérieure à celle permise avec le JPEG pour un même taux de compression.
- Permet également d'appliquer des taux de compression différents sur une même image.
- Certaines parties de l'image peuvent ainsi être plus ou moins compressées en fonction de leur niveau de détail respectif.

Inconvénients :

- JPEG2000 n'est pas compatible avec JPEG.
- L'apparition des zones de flou dans l'image compressée à un taux trop élevé.

Références

- Alistair Moffat et Andrew Thrpin : Compression and coding algorithms, Springer Science+Business Media.
- Claude Berrou : Codes et turbocodes, Springer.
- Didier Le Ruyet et Mylène Pischella : Bases de communications numériques 1: Codage de source et codage de canal, Iste éditions.
- Florent Chavand : Des données à l'information : de l'invention de l'écriture a l'explosion numérique, Iste éditions.
- Hwei P. Hsu Traduit par Bernard Loubières : Communications analogiques et numériques cours et problèmes, Éditeur Chiron Série Schaum
- Jean-Guillaume Dumas, Jean-Louis Roch, Éric Tannier et Sébastien Varrette : THÉORIE DES CODES Compression, cryptage, correction, Dunod, Paris.
- Khalid Sayood : Lossless Compression Handbook-Academic, Elsevier.
- Marie-Pierre Béal et Nicolas Sendrier : Théorie de l'information et codage (Notes de cours).
- Monsef Mekouar : Compression d'images médicales par ondelettes et régions d'intérêt, Thèse de doctorat université du Québec.
- Noureddine Doghmane : Cours codage et compression, Université de Annaba.
- Pascale Jardin : Codage de source, ESIEE (Signaux et Télécommunications).
- Roberto Togneri et Christopher J.S. deSilva : Fundamentals of Information Theory and Coding Design, Chapman and Hall/CRC; 1er edition.