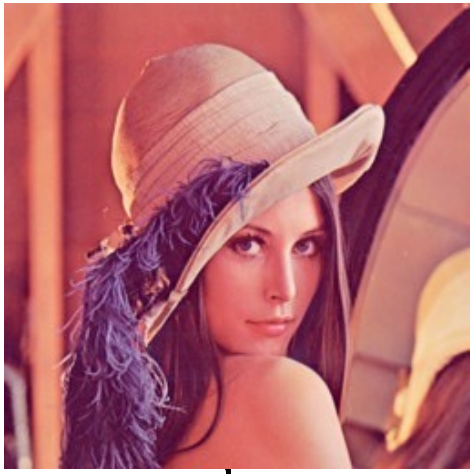
Codage de source



www.lenna.org

I - Introduction

II- Codage sans perte

- Les bases : la théorie de l'information
- Codage d'Huffman
- Codage à base de dictionnaire : Lempel-Ziv (Welch)
- Codage arithmétique

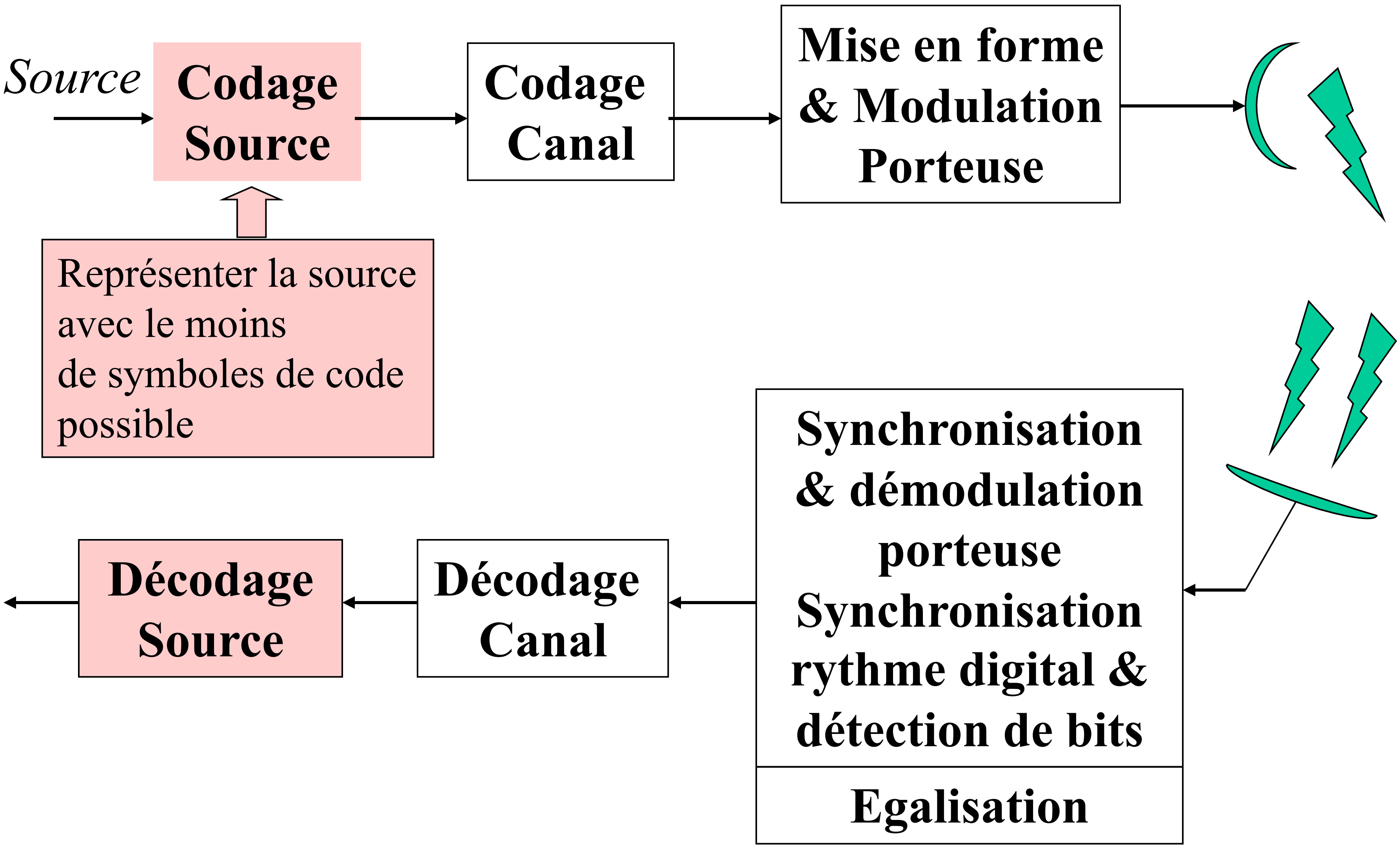
III- Codage avec perte

- Quantification scalaire
- Méthodes prédictives
- Méthodes par transformées

corinne.mailhes@n7.fr

perso.tesa.prd.fr/mailhes

Chaîne de Transmission



Qu'est ce que le codage source ?

• Problème des communications

• Représenter la source avec le moins de symboles de code possible tout en préservant un certain degré de fidélité

• Bases : théorie de l 'information et Shannon

• Chiffres :

➢ étude du sommeil : 1 Go/nuit/patient

➢ parole : 64 kbps

➢ musique stéréo : 1.5 millions de bits / s

➢ CD : 44100 ech / s, 16 bits / éch, 2 canaux

➢ vidéo téléphone : 12 millions de bits / s

➢ vidéo N&B, basse résolution : 8 bits/pixel, 256x256 pixels/trame
24 trames /s

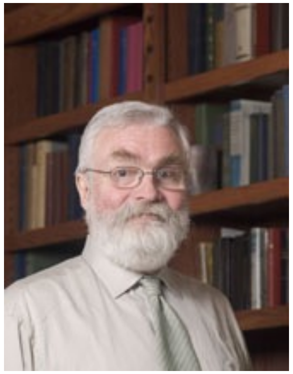
➢ HDTV : 1 billion de bits /s !!!

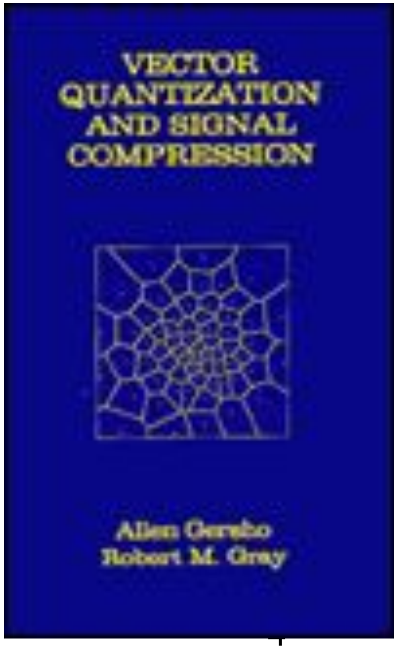
I - Introduction

Codage source = compression

Compression sans distorsion

Compression avec distorsion





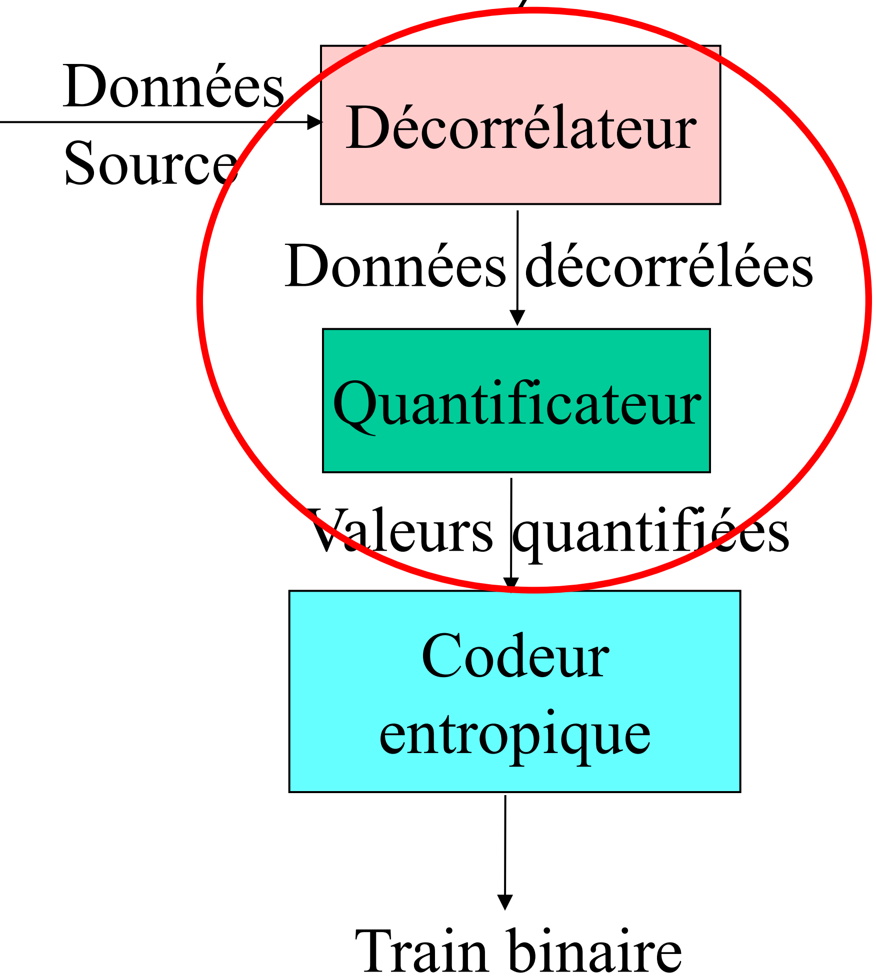
Bibliographie :

A.Gersho, R.M.Gray, « Vector Quantization and Signal Compression », Kluwer Academic, 1991.
D.Salomon, « Data compression, the complete reference », Springer, 1998.
K.Sayood, « Introduction to data compression », Morgan Kaufmann, 1996.
N.Moreau, « Techniques de compression des signaux », Ed Masson, collec. CNET / ENST.
T.M.Cover, J.A.Thomas, «Elements of Information Theory », Wiley Series Telecom, 1991.

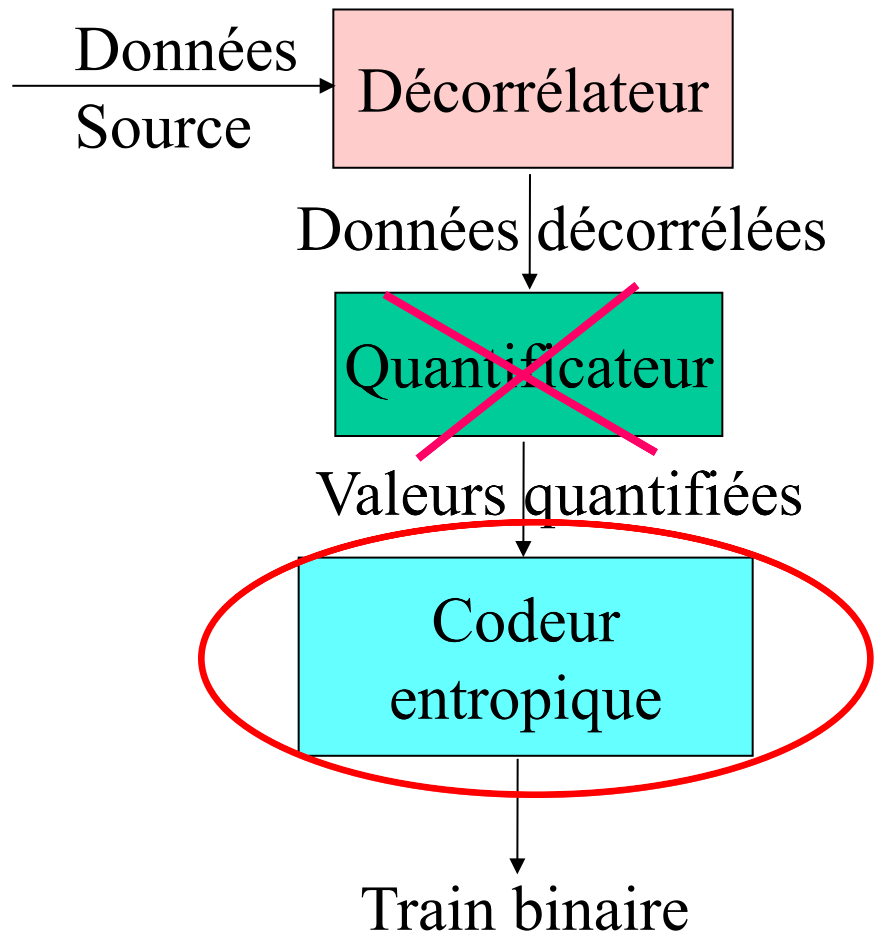
http://ee.stanford.edu/~gray/

Schéma général d'un système de compression

La compression commence par éliminer la redondance



avec distorsion



sans distorsion

Comparer différents codeurs

Evaluation d'un algorithme de codage et comparaison :

- ***taux de bits i.e. taux de compression***
- ***qualité des données reconstruites : notion de distorsion***
- ***complexité de l'algorithme***
- ***retard introduit codage + décodage***
- ***robustesse de l'algorithme aux erreurs de canal et à l'interférence.***

et comparer... sur quelles données ?

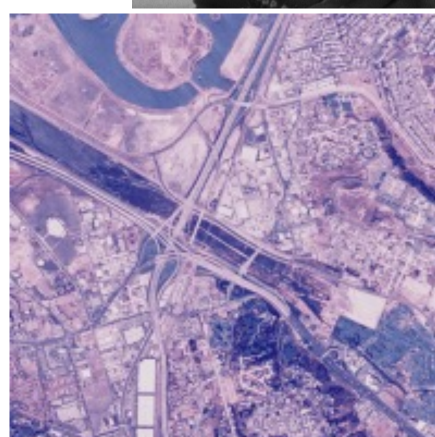
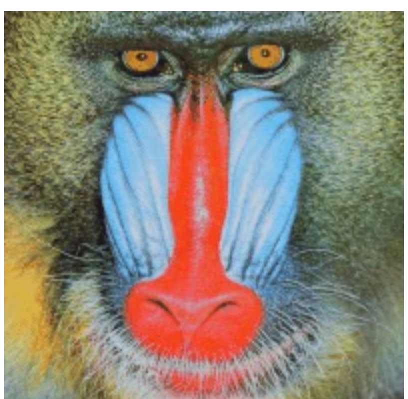
I - Introduction










Codage sans perte
ou codage entropique

Claude Elwood Shannon (1916 – Febr 2001)

1948 les bases : la théorie de l'information

1952 codage d'Huffman

1977 codage à base de dictionnaire : Lempel-Ziv (Welch)

1981 codage arithmétique

C. E. Shannon, « A mathematical theory of communication »,
Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, July and October, 1948.
Disponible sur le web

Les bases : la théorie de l'information

Les principes énoncés par Shannon :

Comment définir la quantité d'information d'un message ?

Source émettant un message = source aléatoire

Sans aléatoire, pas d'information !

Quantité d'information d'un message :

$i(x_k) = - \log2 (p_k)$ bits

p_k étant la probabilité d'émission du message x_k

Entropie d'une source X

= quantité d'information moyenne émise par la source

= quantité de doute moyen de la source

= mesure de son imprédictibilité

$H(X) = - \sum_k \; p_k \, log_2 \, (p_k)$ (bits)

Codage sans distorsion

Borne inférieure de la longueur moyenne d'un code avec D symboles

Comment construire un code irréductible* tel que :

$$H(X) / \log_2 (D) \leq \bar{n} \leq H(X) / \log_2 (D) + 1$$

$H(X) = - \sum_k p_k \log_2 (p_k)$ (bits) entropie de la source

$\bar{n} = \sum_k p_k n_k$ longueur moyenne du code

Principes issus de la théorie de l'information :

1. Faire du codage à longueur variable
2. Coder des blocs de messages

- 2 approches générales :

- Par la statistique : Huffman, arithmétique
- À l'aide d'un dictionnaire

(*) Irréductible : code à longueur variable pouvant être déchiffré instantanément, sans ambiguïté (aucun mot n'est le début d'un autre)

Exemple d’un code à longueur variable

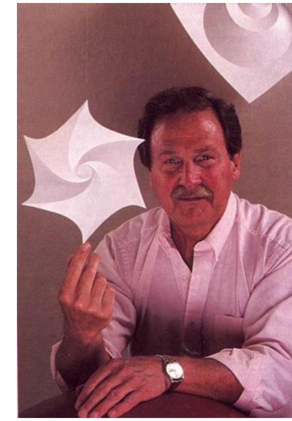
INTERNATIONAL MORSE CODE

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to five dots.

A	• —
B	— • • •
C	— • — •
D	— • •
E	•
F	• • — •
G	— — •
H	• • • •
I	• •
J	• — — —
K	— • —
L	• — • •
M	— —
N	— •
O	— — —
P	• — — •
Q	— — • —
R	• — •
S	• • •
T	—
U	• • —
V	• • • —
W	• — —
X	— • • —
Y	— • — —
Z	— — • •
1	• — — — —
2	• • — — —
3	• • • — —
4	• • • • —
5	• • • • •
6	— • • • •
7	— — • • •
8	— — — • •
9	— — — — •
0	— — — — —

II- Codage sans perte

Code d'Huffman



Code d'Huffman (1952) : codage optimal

David A. Huffman (9 août 1925 - 7 octobre 1999)

D.A. Huffman, "A method for the construction of minimum-redundancy codes", Proceedings of the I.R.E., sept 1952, pp 1098-1102

http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf

Cas binaire :

1. on classe les messages par ordre de probabilité décroissante,
2. on regroupe les 2 messages les moins probables, en leur affectant une probabilité égale à la somme des probabilités. Les 2 messages auront le même code sauf la fin : le 1er se verra affecter un symbole « 0 » et le 2ème un symbole « 1 »,
3. on refait 1 jusqu 'à épuisement.
4. La lecture des mots-code se fait en lisant le tableau ainsi constitué de gauche à droite : on lit les mots-code à l'envers (de la fin vers le début)

Cas D différent de 2 :

On fait de même en remplaçant « 2 » par « D » et les symboles 0 et 1 par $u_1,\ldots,u_D$

☠💣☠ Initialisation : on regroupe non pas D symboles à la 1ère itération mais :

2 + reste de la division de N-2 par D-1

Code d'Huffman : un exemple

Mots-code	Messages de la source	Probas													
1	A	0.4		0.4		0.4		0.4		0.4		0.4		0.6	0
001	B	0.18		0.18		0.18		0.19		0.23		0.37	0	0.4	1
011	C	0.1		0.1		0.13		0.18		0.19	0	0.23	1		
0000	D	0.1		0.1		0.1		0.13	0	0.18	1				
0100	E	0.07		0.09		0.1	0	0.1	1						
0101	F	0.06		0.07	0	0.09	1								
00010	G	0.05	0	0.06	1										
00011	H	0.04	1												

Longueur moyenne atteinte par Huffman : 2.61
H(X)=2.55 bits soit une efficacité de E=97.8%

Code d’Huffman

Il existe trois variantes de l'algorithme de Huffman :

- statique : probabilités fixées au départ. La table de probas connue du décodeur par défaut.
- semi-adaptatif : algorithme à 2 passes.
  1- Le fichier est d'abord lu, de manière à estimer les probas, le code est construit,
  2- Codage

Il sera nécessaire pour la décompression de transmettre l'arbre.

- adaptatif : méthode offrant a priori les meilleurs taux de compression :

les probas sont mises à jour de manière dynamique au fur et à mesure de la compression.
Gros désavantage : devoir reconstruire le code à chaque fois,
ce qui implique un temps d'exécution énorme.

Faller (1973), Gallagher (1978) prennent en compte statistique
des messages déjà rencontrés
Knuth (1985), Vitter (1987) améliorent

Machine de Rice : plusieurs codes, prendre le meilleur

4 codes d’Huffman sur Voyager 1978-1990
32 codes d’Huffman dans MPEG Audio layer 3

II- Codage sans perte

Comment se rapprocher de la limite inférieure ?

En s'appuyant sur principe énoncé par Shannon

$$H(X) \,/\, \mathit{log}\mathbf{2}\,(\,D) \leq \overline{n}$$

Théorème de Shannon *: il faut coder les messages*
Non pas individuellement mais par blocs de plusieurs messages
Plus on groupe des messages, plus on se rapproche de la limite…

p	o	u	r		a	u	g	m	e	n	t	e	r		l		e	f	f	i	c	a	c	i	t	e		i	l		f	a	u	t		c	o	d

Mais avec Huffman, cela augmente la complexité du code (dictionnaire !!)

II- Codage sans perte

Si les probabilités sont inconnues, Huffman n'est plus optimal…

Mieux qu'Huffman
sur données ASCII :
Codes à base de dictionnaire

Lempel-Ziv (Welch)
(1977-1978, 1984)

les poules du couvent couvent…

LZ78 (LZW)

LZ77 :
PKZIP
Lharc
ARJ

LZ78 :
COMPRESS
ARC

Codeur et décodeur au départ connaissent le code ASCII (8bits)

Code ASCII :

0 : _

:

255 : ÿ

On prévoit d'étendre le dictionnaire sur 9 bits par exemple,

Etapes de l'émission :

« l » : on envoie son code « 108 »

« e » : on envoie son code « 101 » et on rajoute dans la table de code : **256 : le**

« s » : on envoie son code « 115 » et on rajoute dans la table de code : **257 : es**

« _ » : on envoie son code « 0 » et on rajoute dans la table de code : **258 : s_**

« p » : on envoie son code « 112 » et on rajoute dans la table de code : **259 : _p**

« o » : on envoie son code « 111 » et on rajoute dans la table de code : **260 : po**

« u » : on envoie son code « 117 » et on rajoute dans la table de code : **261 : ou**

« le » : on envoie son code « 256 » et on rajoute dans la table de code : **262 : ule**

« s_ » : on envoie son code « 258 » et on rajoute dans la table de code : **263 : les_**

« d » : on envoie son code « 100 » et on rajoute dans la table de code : **264 : s_d**

etc…

le décodeur reçoit et enrichit sa table au fur-et-à-mesure...

II- Codage sans perte

Mieux qu'Huffman (mais plus cher !)
Codage arithmétique

JBIG
standart
Joint
Bi-level
Image
Compression
par ex.
fax

Associer au message (ou à un flot de messages)
un nombre compris entre 0 et 1
en tenant compte de la statistique de la source.

Huffman :1 symbole = 1 mot-code
Arithmétique : 1 flot de symboles = nbre en virgule flottante
mais coûteux en calculs !

Algorithmes supplémentaires

Run-length coding (RLE) : si un message apparaît *n* fois consécutives,
Au lieu de coder x x x x…x *n* fois, on code : n x
Ou plutôt pour prévenir qu'il s'agit de RLE : @ n x

Flag de RLE

Chaîne de N caractères avec M répétitions de longueur L moyenne chacune
Chaque répétition est remplacée par 3 caractères
Taux de compression : N / (N – ML + 3M)

Associer au message (ou à un flot de messages) un nombre compris entre 0 et 1 en tenant compte de la statistique de la source.

Codage : $x_{inf}(n) < x < x_{sup}(n)$ *avec* $x_{inf}(n) = x_{inf}(n\text{-}1) + p_{inf}\%\,(x_{sup}(n\text{-}1) - x_{inf}(n\text{-}1))$

et $x_{sup}(n) = x_{inf}(n\text{-}1) + p_{sup}\%\,(x_{sup}(n\text{-}1) - x_{inf}(n\text{-}1))$

JBIG
standart
Joint
Bi-level
Image
Compression
par ex.
fax

$[p_{inf}\%, p_{sup}\%]$: intervalle associé à la lettre en fonction de sa statistique

Dans l'exemple : ***les_poules***

	_	e	l	o	p	s	u
$p_{inf}\%$:	0	0.1	0.3	0.5	0.6	0.7	0.9
$p_{sup}\%$:	0.1	0.3	0.5	0.6	0.7	0.9	1

Codage :

« l » : 0.3 < x < 0.5
« e » : 0.32 < x < 0.36 (10% et 30% de l'intervalle [0.3,0.5])
« s » : 0.348 < x < 0.356 (70% et 90% de l'intervalle [0.32,0.36])
« _ » : 0.3480 < x < 0.3488
« p » : 0.34848 < x < 0.34856
« o » : 0.34852 < x < 0.348528
« u » : 0.3485272 < x < 0.348528
« l » : 0.34852744 < x < 0.3485276
« e » : 0.348527456 < x < 0.348527488
« s » : 0.3485274784 < x < 0.3485274848
3485274784 code le message (compris entre 2^31 et 2^32) : 32 bits sont nécessaires
(code ASCII : 10*8=80 bits, entropie = 2.72 bits)

Décodage : $p_{inf}\% < x(n) < p_{sup}\%$: on décode la lettre correspondante

et $x(n+1) = (x(n) - p_{inf}\%) / (p_{sup}\% - p_{inf}\%)$

algorithmes supplémentaires

Run-length coding (RLE) : si un message apparaît *n* fois consécutives,
Au lieu de coder x x x x…x *n* fois, on code : n x
Ou plutôt pour prévenir qu'il s'agit de RLE : @ n x

Flag de RLE

Chaîne de N caractères avec M répétitions de longueur L moyenne chacune
Chaque répétition est remplacée par 3 caractères
Taux de compression : N / (N – ML + 3M)

Move to front coding : pour données non numériques

BZIP2 (Burrows-Wheeler)

Exemple avec 4 caractères a,b,c,d à coder 0,1,2,3

a b a b c c b b c c c c b d b c c

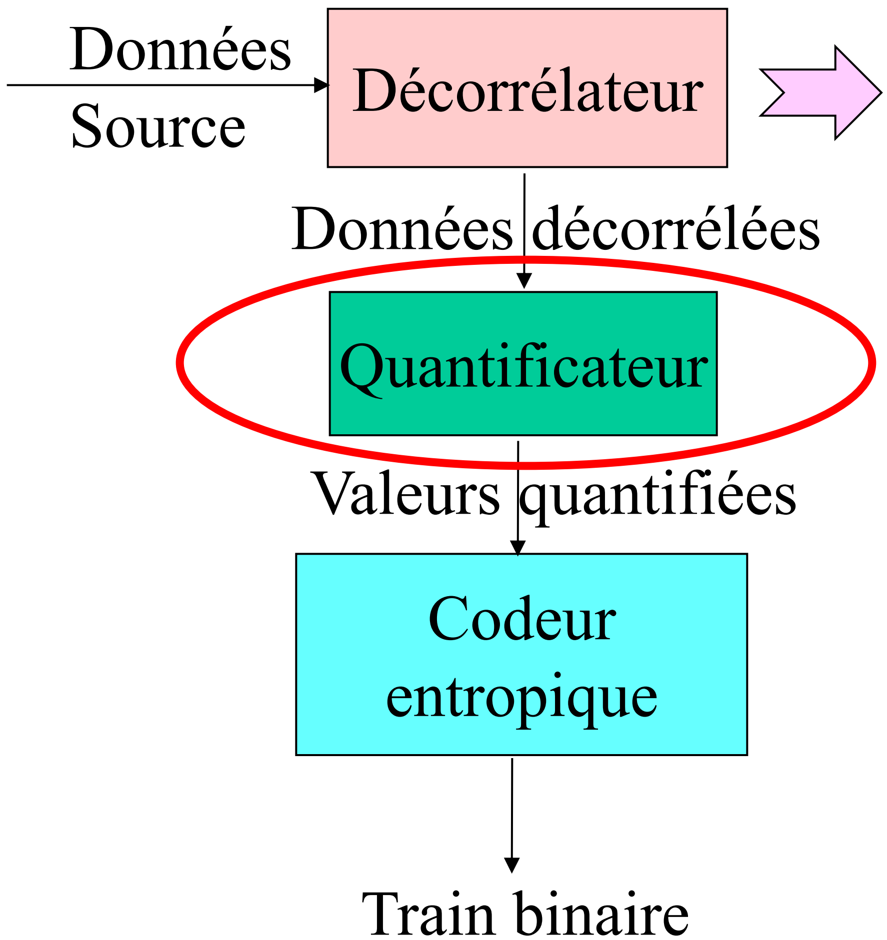
« a » on code « 0 » avec table de codage : a->0, b->1, c->2, d->3
« b » : on code 1, table de codage se modifie : b->0, a->1, c->2, d->3
« a » : on code 1, table de codage se modifie : a->0, b->1, c->2, d->3
« b » : on code 1, table de codage se modifie : b->0, a->1, c->2, d->3
« c » : on code 2, table de codage se modifie : c->0, b->1, a->2, d->3
« c » : on code 0 etc…

Modifie les fréquences des symboles, faire ensuite code entropique

Nécessité du Codage avec perte

Chiffres :

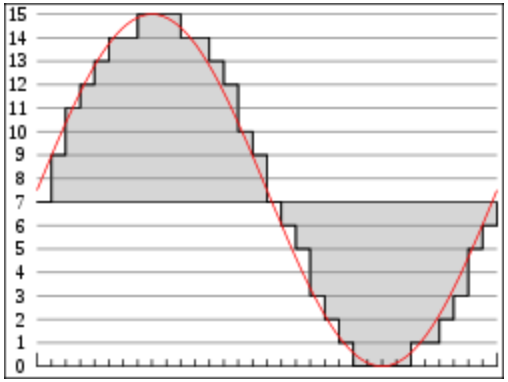
- parole : 64 kbps
- musique stéréo : 1.5 millions de bits / s
- HDTV : 1 billion de bits /s !!!



méthodes prédictives,
méthodes par transformées,
quantification vectorielle
ET méthodes hybrides

III- Codage avec perte : quantification scalaire

Ce qu'il faut savoir…



Quantification uniforme :
pas de quantification Δ constant,

si
N nombre de niveaux de quantification $=2^n$ suffisamment grand
(n nombre de bits)

SNR $\approx 6n - 20\log(a) + 10.8$ dB avec $a = B/\sigma_x$ (B dynamique du Q)
rapport signal à bruit = fonction linéaire du nombre de bits (*vu en 1SN*)
+ 1 bit = + 6 dB

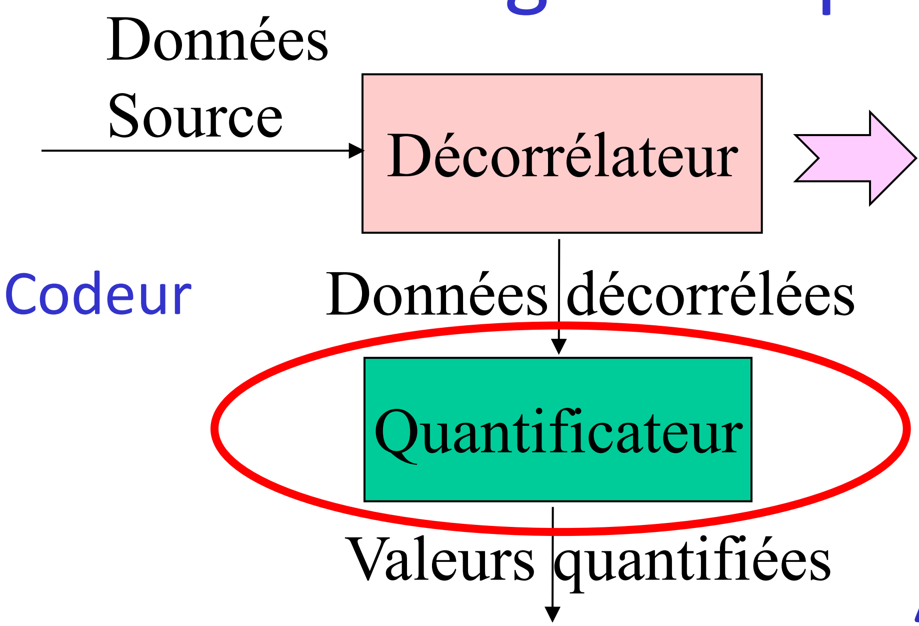
Ou vu autrement :

$$\sigma^2_\varepsilon = \sigma^2_x \, a^2 \, 2^{-2n} / 12$$

La puissance de l'erreur de quantification est proportionnelle à la puissance du signal à l'entrée du quantificateur

Quantification non uniforme : en téléphonie, lois A (Europe) et μ (Amérique du Nord et Japon) G711 : Q. logarithmique

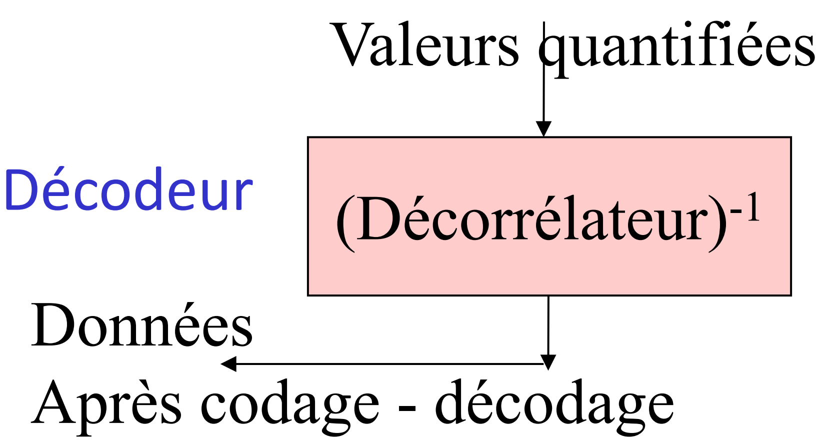
Codage avec perte



Rôle du « décorrélateur » :

- Eliminer la redondance (principe de base de la compression)
- Transformer les données pour être appropriées à la quantification (= compression)

La puissance de l'erreur de quantification est proportionnelle à la puissance du signal à l'entrée du quantificateur



Maitriser la distorsion :

- Distorsion faite au quantificateur
- Egale à celle entre données source et données après codage-décodage

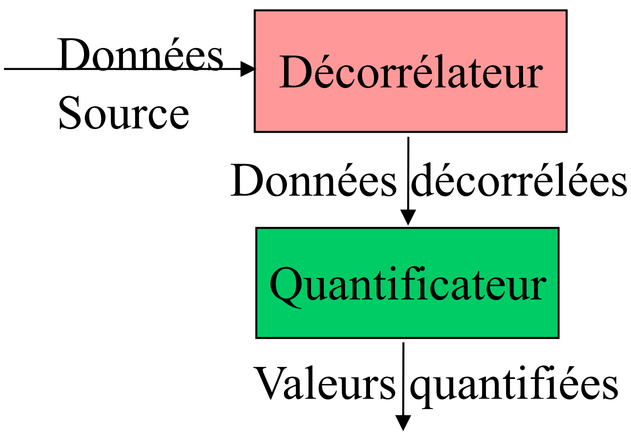
III- Codage avec perte : méthodes prédictives

Idée : Coder les différences entre échantillons plutôt que les échantillons

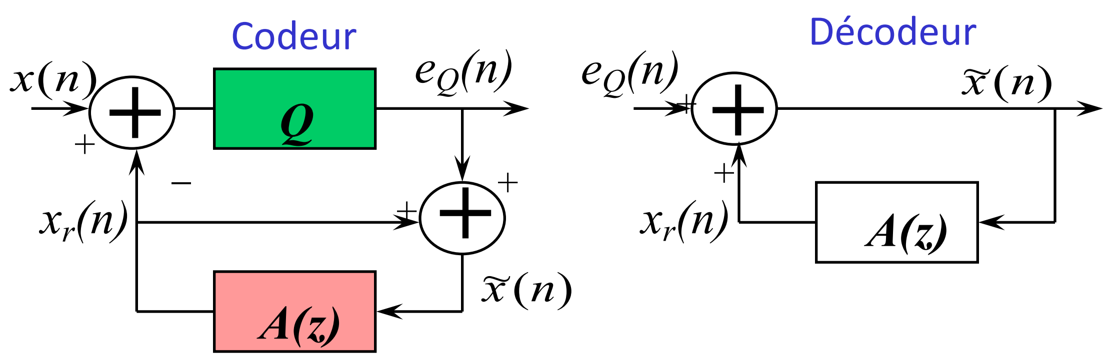
mieux : coder une différence **pondérée** dont la puissance est plus faible que celle du signal !

$e(n) = x(n) - \sum_k a_k\, x(n-k)$ a_k minimisent la puissance de $e(n)$

On code $e(n)$, l'Erreur de Prédiction Linéaire (EPL), l'erreur de modèle, la partie non prédictible



Differential Pulse Code Modulation (DPCM)



Gain constant par rapport au PCM : $SNR_{DPCM} = SNR_{QS} + 10 \log G_p$ avec $G_p = \sigma^2_x / \sigma^2_e$

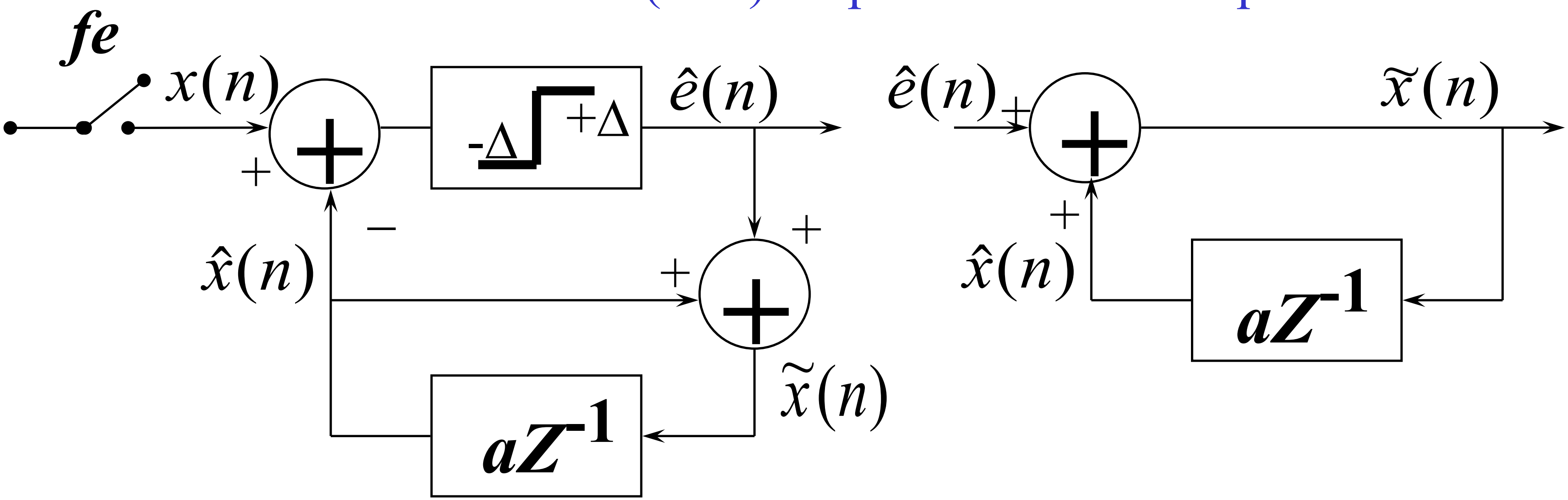
1972 : G.711 : PCM - 64 kbits/s :
8 bit-quantizer
after non linear compression

Versions adaptatives : ADPCM

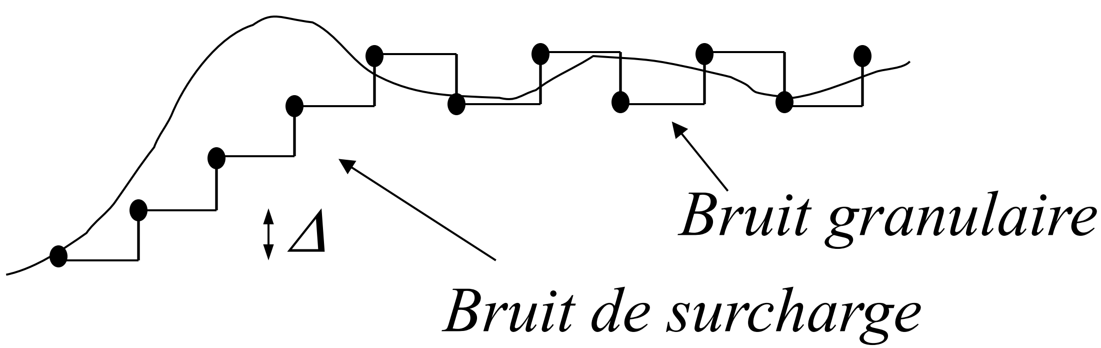
1984 : G721 : ADPCM - 32 kbits /s
4 bit adaptive quantizer
adaptive predicter, MOS ≈ 4

Version économique :

Modulation Delta (DM) ou plutôt version adaptative ADM

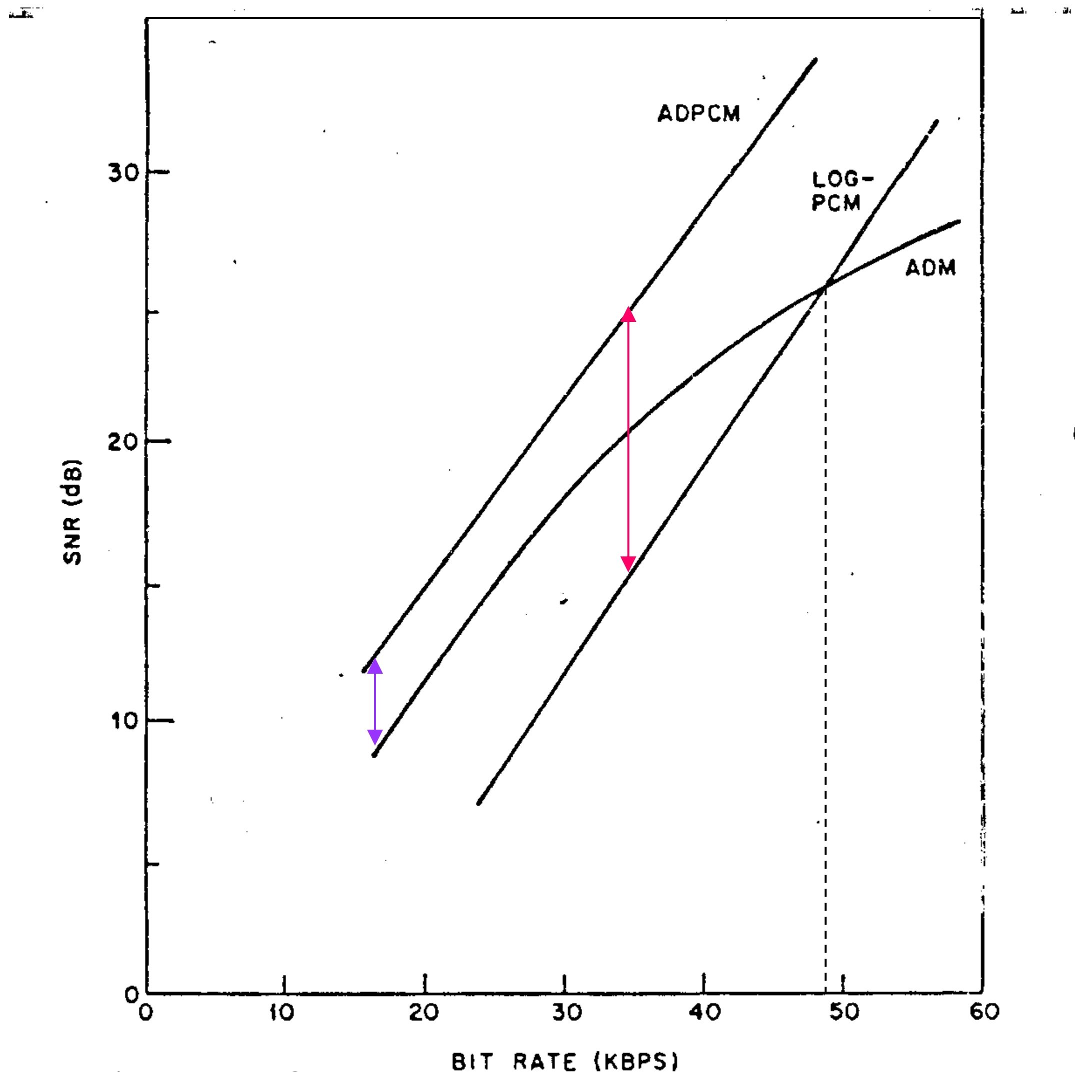


$$\tilde{x}(n) = a\tilde{x}(n\text{-}1) + \Delta\, sign(x(n) - a\tilde{x}(n\text{-}1)) \quad \text{avec } a = r_1 \text{ proche de } 1$$



Pas de Quantif. à rendre **adaptatif**

III- Codage avec perte : méthodes prédictives

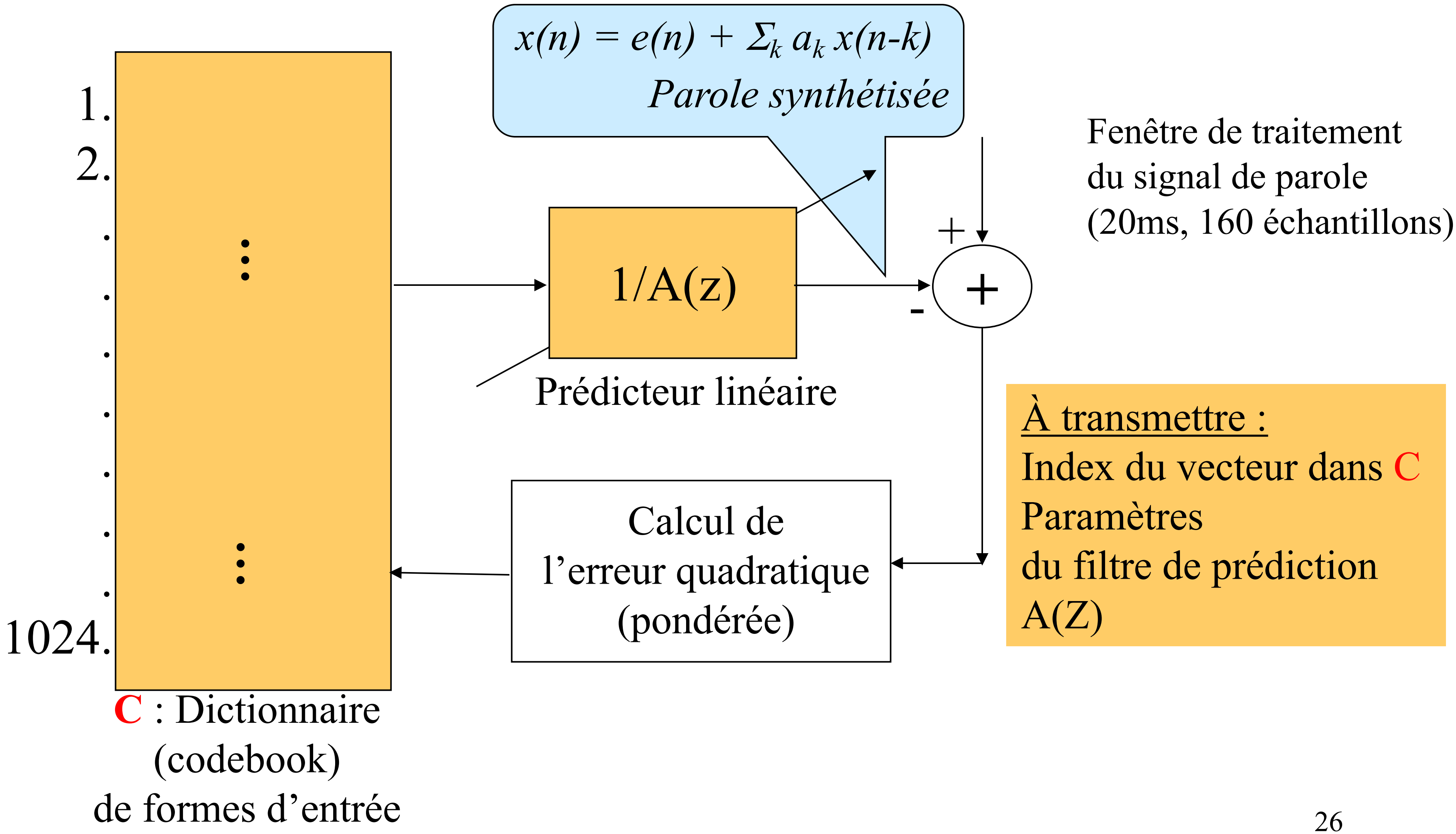


Comparaison des performances Log-PCM, ADM et ADPCM en termes de taux de bits et de SNR

http://mailhes.perso.enseeiht.fr/compression/index.html

Application au codage de la parole : les codeurs CELP

Code Excited Linear Predictor

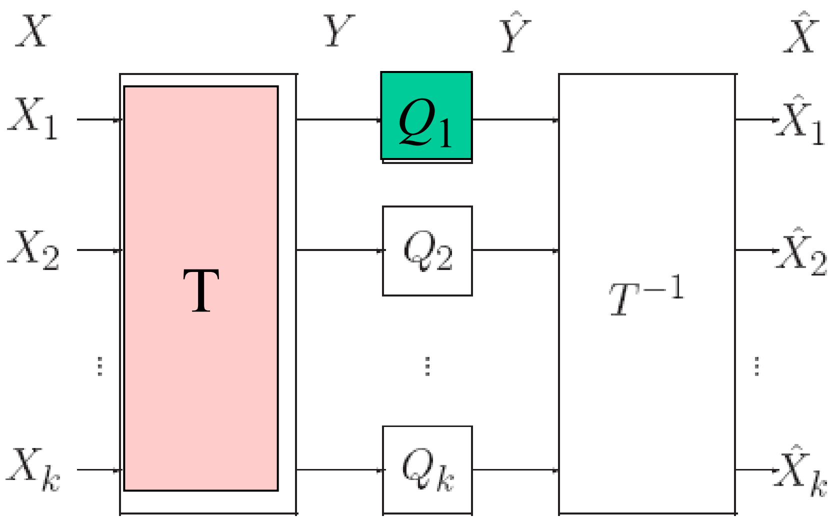


Transform Coding

Mathews and Kramer (1956)
Huang, Habibi, Chen (1960s)

Dominant image coding (lossy compression) method: ITU and other standards p*64, H.261, H.263, JPEG, MPEG C-Cubed, PictureTel, CLI

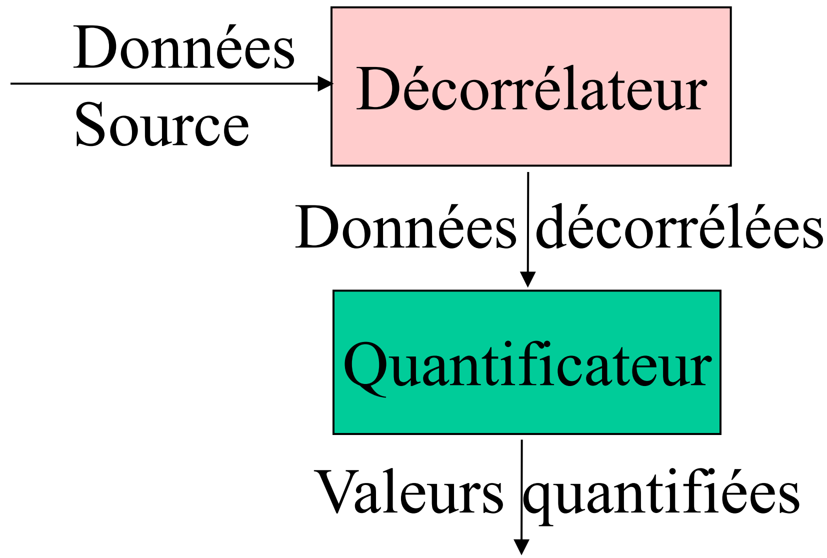
Codage par Transformée : le principe de base



Transform Coding

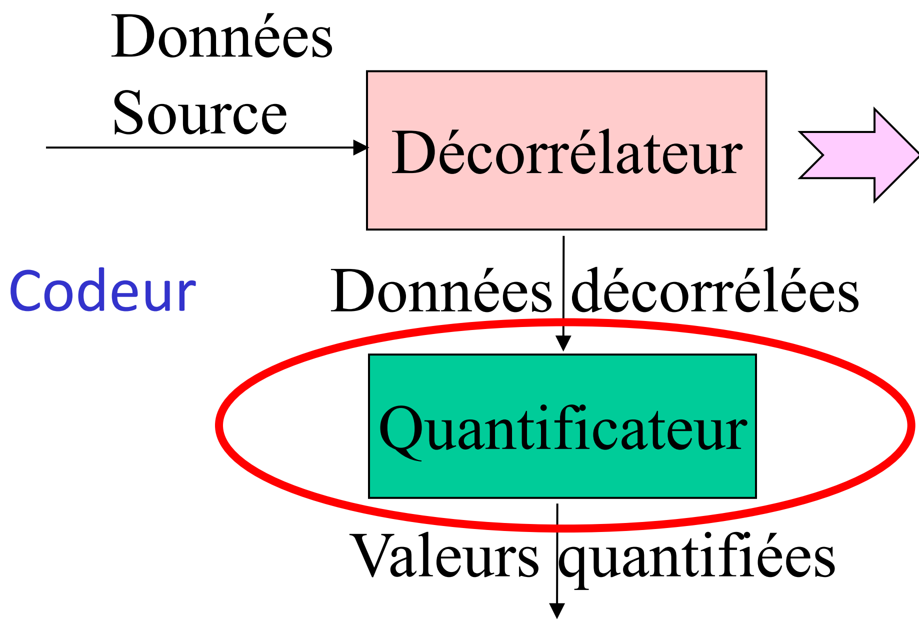
$$X = (X_1, X_2, \ldots, X_k)^t$$

$$Y = T(X), \hat{Y} = Q(Y), \hat{X} = T^{-1}(\hat{Y})$$



III- Codage avec perte : méthodes par transformées

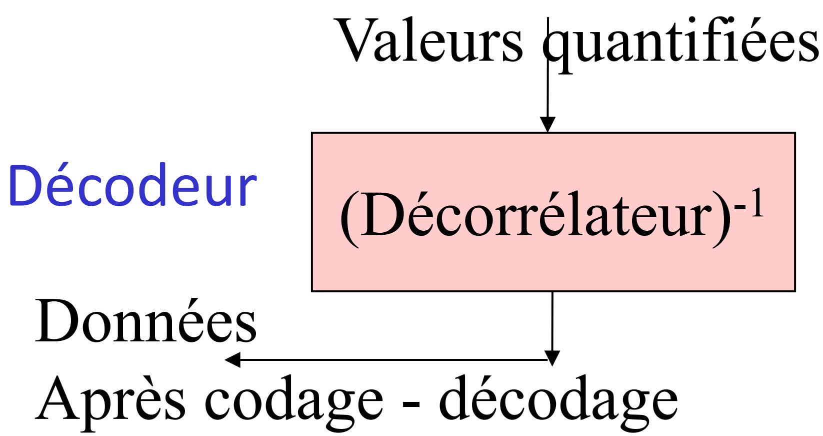
Quelle transformée ?



Rôle du « décorrélateur » :

- Eliminer la redondance (principe de base de la compression)
- Transformer les données pour être appropriées à la quantification (= compression)

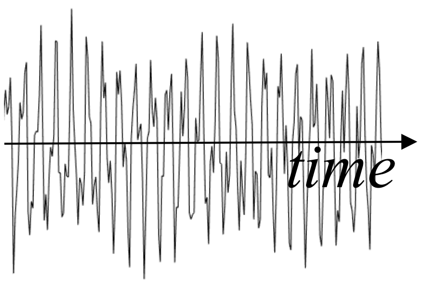
La puissance de l'erreur de quantification est proportionnelle à la puissance du signal à l'entrée du quantificateur



Maitriser la distorsion :

- Distorsion faite au quantificateur
- Egale à celle entre données source et données après codage-décodage

Exemple d'intérêt d'une transformée pour la compression



DCT
FFT
Wavelet..

250 samples,
Each sample on B bits
= 250 B bits

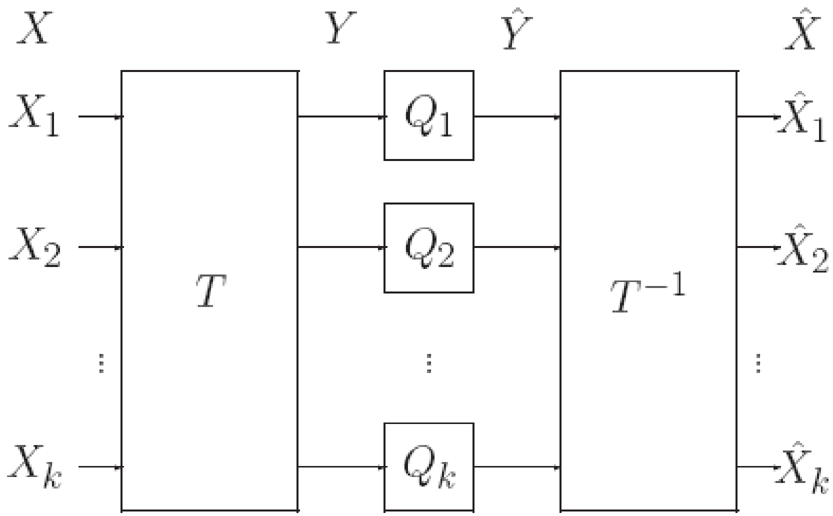
Ici seulement
une dizaine de coefficients non nuls
= 10 B bits nécessaires pour coder

obligatoire

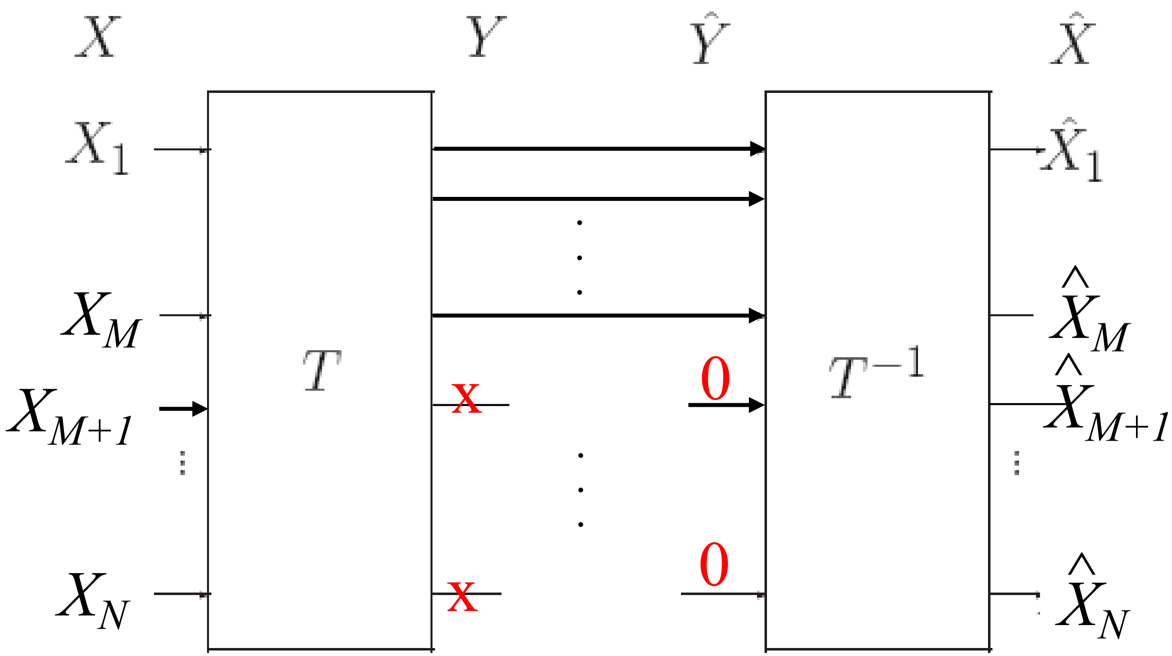
Quelle transformée ?
Unitaire : erreur entre Y et $\hat{Y}$ de même puissance qu'entre X et $\hat{X}$

recommandé

Quelle transformée unitaire ?
- qui décorrèle
- qui concentre le maximum d'énergie sur le minimum de coefficients transformés



Codage par transformée : implantation simple
(sans choix de quantificateur)



Karhunen-Loeve Transform

Often referred to as "optimal transform"

Idea: Decorrelate components of vector X. If also Gaussian, makes independent.

Consider 1D case, assume real:

$R_X = E[XX^t]$
$X \to Y = TX$ with $R_Y = E[YY^t]$ diagonal.

u_i denote the eigenvectors of R_X (normalized to unit norm)
λ_i the corresponding eigenvalues (ordered)

The Karhunen-Loeve transform matrix is then defined as $T = U^t$ where

$$U = [u_i u_2 \ldots u_k],$$

(the columns of U are the eigenvectors of R_X)

Then

$$R_Y = E[YY^t] = E[U^t X X^t U] = U^t R_X U = diag(\lambda_i).$$

Note that the variances of the transform coefficients are the eigenvalues of the autocorrelation matrix R_X.

III- Codage avec perte : méthodes par transformées

Quelle transformée unitaire choisir ?

Celle qui décorrèle et qui concentre le mieux l'énergie

TABLE 5.3 Summary of Image Transforms

DFT/unitary DFT	Fast transform, most useful in digital signal processing, convolution, digital filtering, analysis of circulant and Toeplitz systems. Requires complex arithmetic. Has very good energy compaction for images.
Cosine	Fast transform, requires real operations, near optimal substitute for the KL transform of highly correlated images. Useful in designing transform coders and Wiener filters for images. Has excellent energy compaction for images.
Sine	About twice as fast as the fast cosine transform, symmetric, requires real operations; yields fast KL transform algorithm which yields recursive block processing algorithms, for coding, filtering, and so on; useful in estimating performance bounds of many image processing problems. Energy compaction for images is very good.
Hadamard	Faster than sinusoidal transforms, since no multiplications are required; useful in digital hardware implementations of image processing algorithms. Easy to simulate but difficult to analyze. Applications in image data compression, filtering, and design of codes. Has good energy compaction for images.
Haar	Very fast transform. Useful in feature extracton, image coding, and image analysis problems. Energy compaction is fair.
Slant	Fast transform. Has "image-like basis"; useful in image coding. Has very good energy compaction for images.
Karhunen-Loeve	Is optimal in many ways; has no fast algorithm; useful in performance evaluation and for finding performance bounds. Useful for small size vectors e.g., color multispectral or other feature vectors. Has the best energy compaction in the mean square sense over an ensemble.
Fast KL	Useful for designing fast, recursive-block processing techniques, including adaptive techniques. Its performance is better than independent block-by-block processing techniques.
Sinusoidal transforms	Many members have fast implementation, useful in finding practical substitutes for the KL transform, analysis of Toeplitz systems, mathematical modeling of signals. Energy compaction for the optimum-fast transform is excellent.
SVD transform	Best energy-packing efficiency for any given image. Varies drastically from image to image; has no fast algorithm or a reasonable fast transform substitute; useful in design of separable FIR filters, finding least squares and minimum norm solutions of linear equations, finding rank of large matrices, and so on. Potential image processing applications are in image restoration, power spectrum estimation and data compression.

Quelle transformée unitaire ?

Solution optimale = Transformée de Karhunen-Loeve (KL)
Mais matrice T dépend de la corrélation du signal…pas commode !

Deux transformées dominent pour leurs performances :

1. La DCT (Discrete Cosine Transform)
2. La DWT (Discrete Wavelet Transform)

Intérêt de la DCT : simple à implanter,
Schéma sans quantification possible (tout ou rien)
Base du standard JPEG 92-93

Quelle transformée ?

Many other transforms, but dominant ones are discrete cosine transform (DCT, the 800 lb gorilla) and discrete wavelet transform (DWT).

Theorems to show that if image $\approx$ Markovian, then DCT approximates KL

Wavelet fans argue that wavelet transforms $\approx$ decorrelate wider class of images.

Often claimed KL "optimal" for Xform codes

Only true (approximately) if high rate, optimal bit allocation among coefficients, and Gaussian.

III- Codage avec perte : méthodes par transformées

Codage par transformée :
Implantation simplifiée :
en ne transmettant que
M coefs sur les N coefs
transformés

TC = N/M

DCT :
Discrete Cosine Transform

The 1-D discrete cosine transform is defined by

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

Hadamard transform :
Matrice de +1 et -1

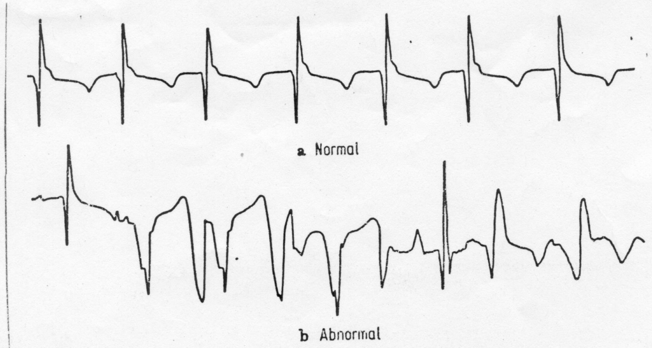


Fig. 9.7 Original ECG's using all the 128 data points

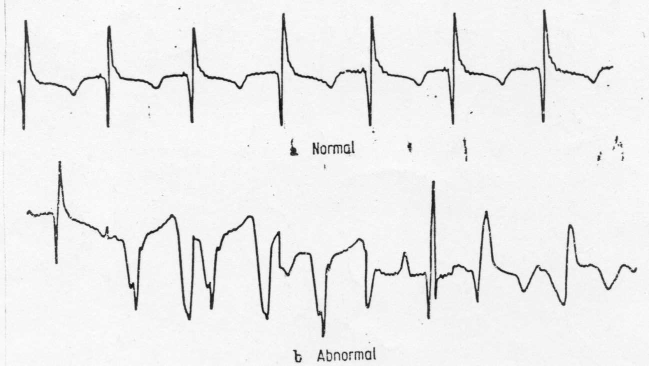


Fig. 9.8 Reconstructed ECG's using the dominant 43 DCT components

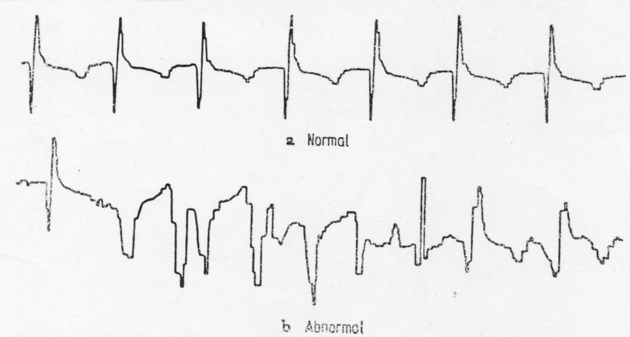


Fig. 9.9 Reconstructed ECG's using the dominant 43 HT components

Variance des composantes du vecteur transformé (dimension 16)

(signal théorique processus de Markov) pour différentes transformées

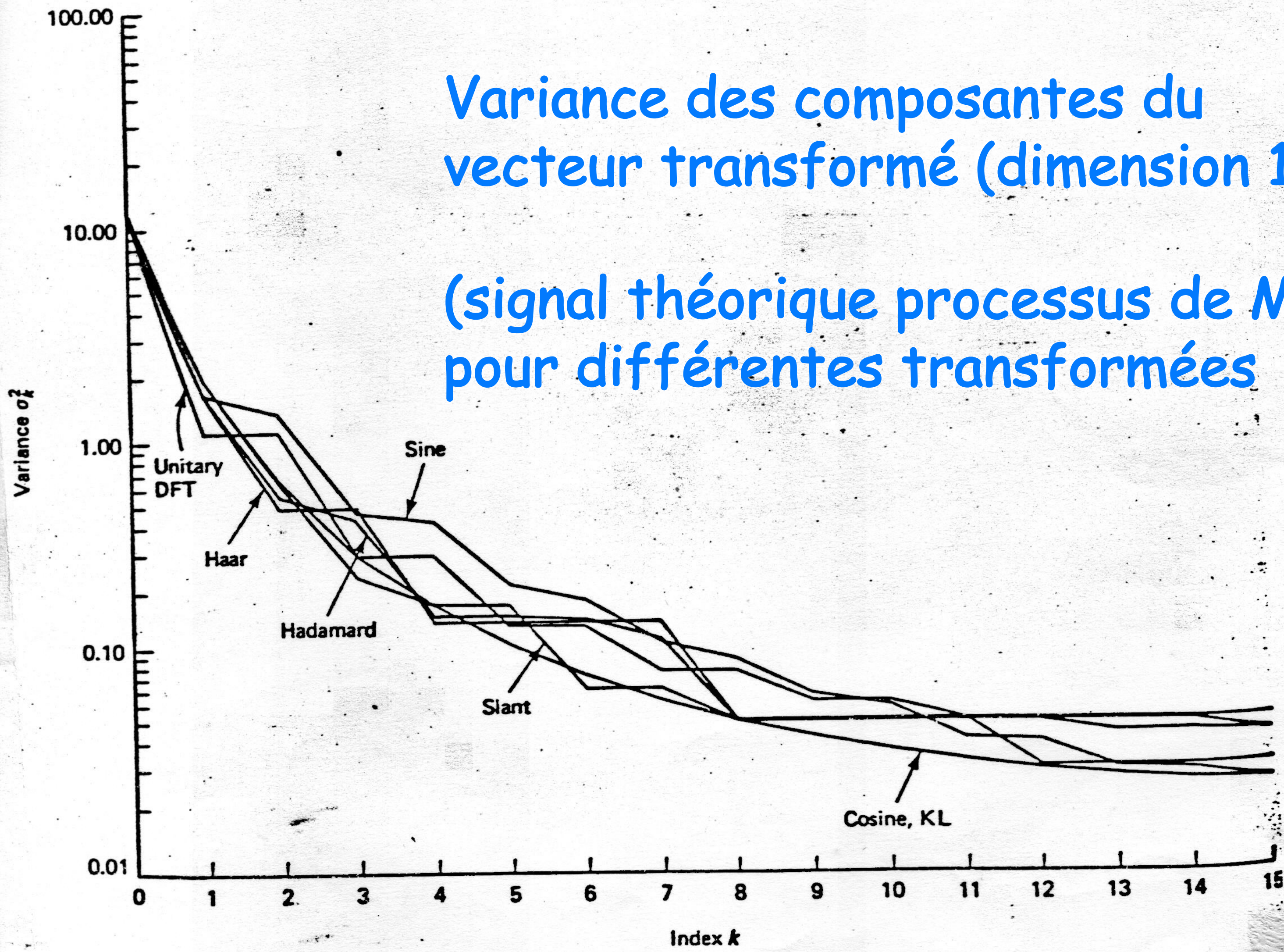
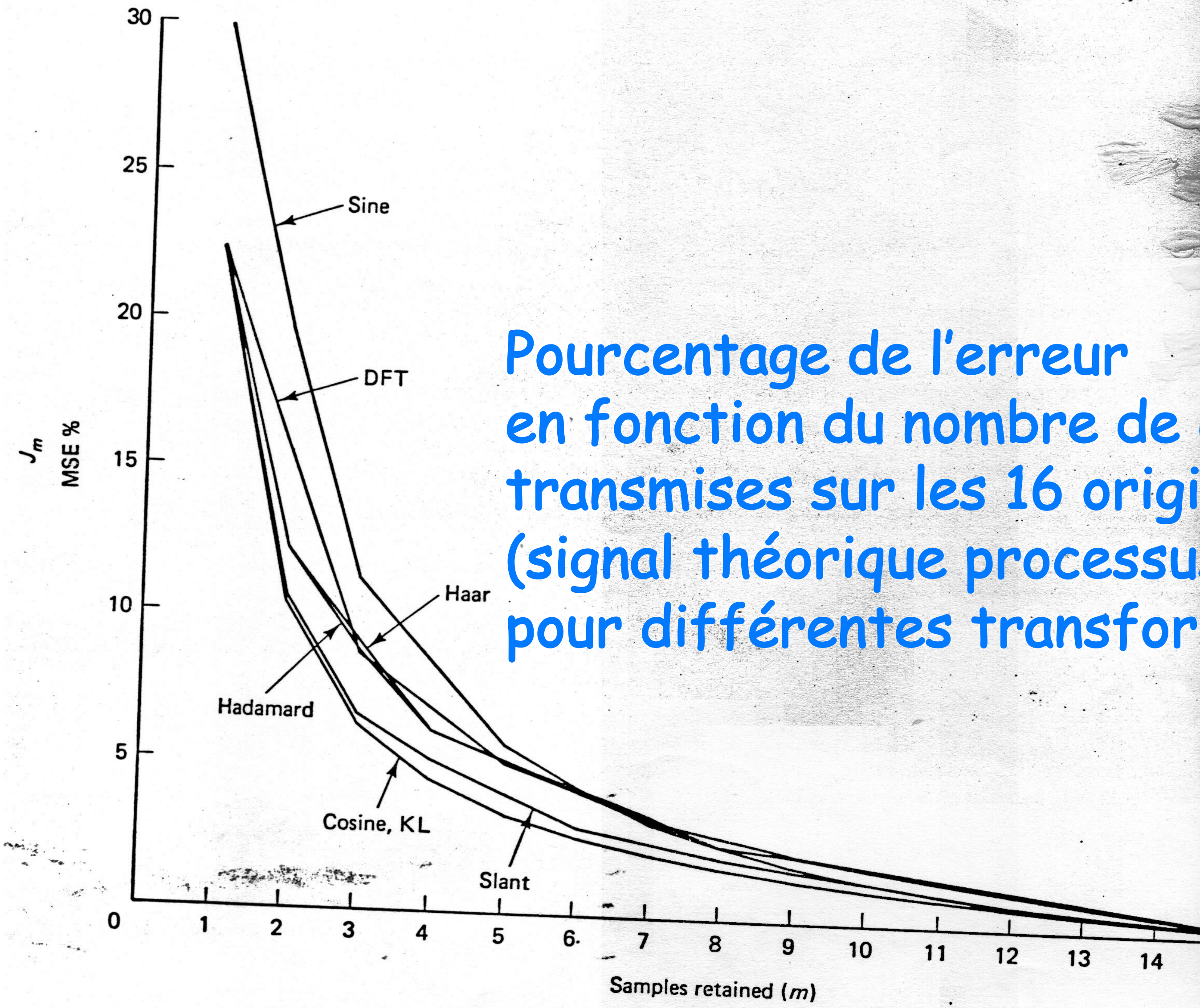
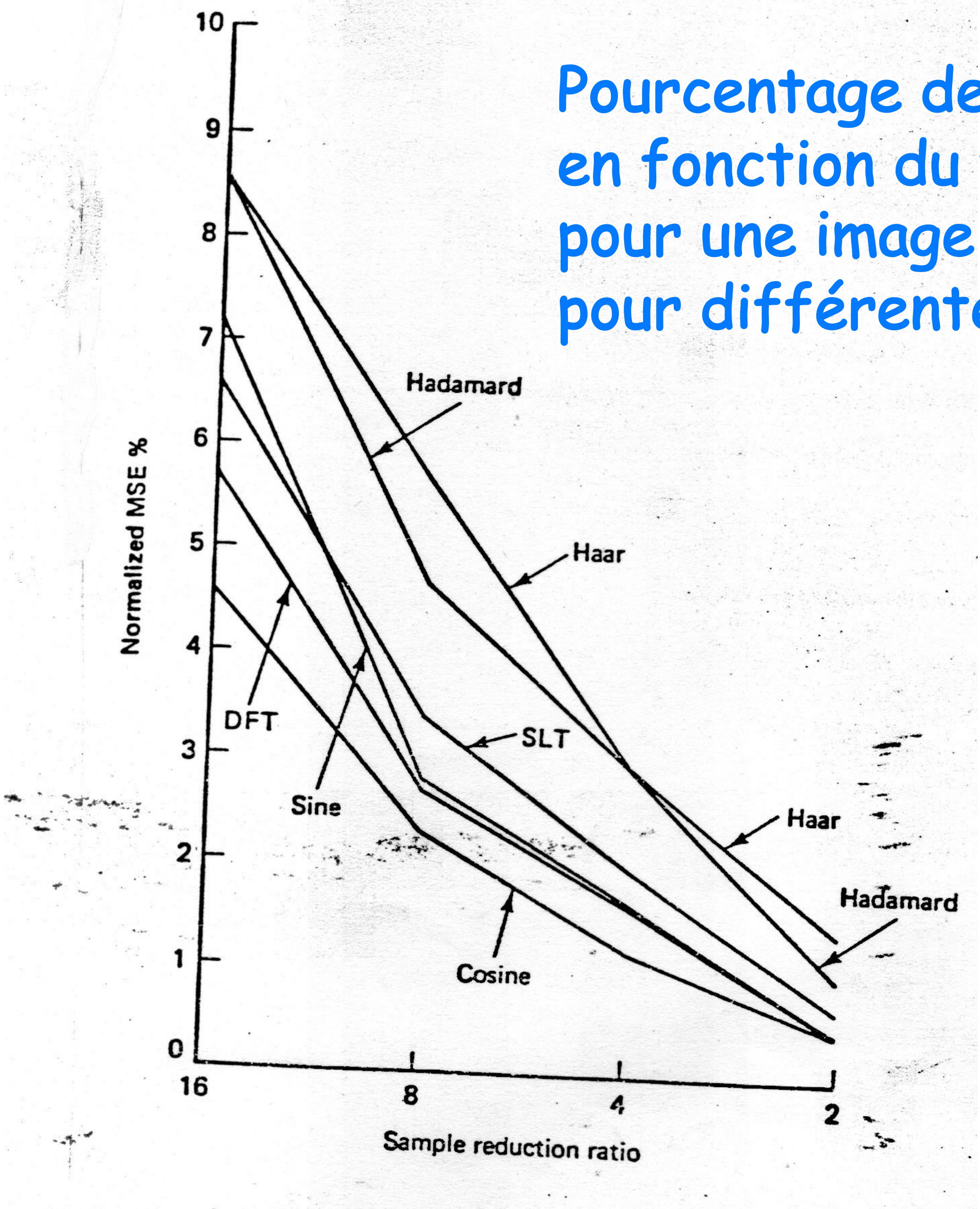


Figure 5.18 Distribution of variances of the transform coefficients (in decreasing order) of a stationary Markov sequence with $N = 16$, $\rho = 0.95$ (see Example 5.9).



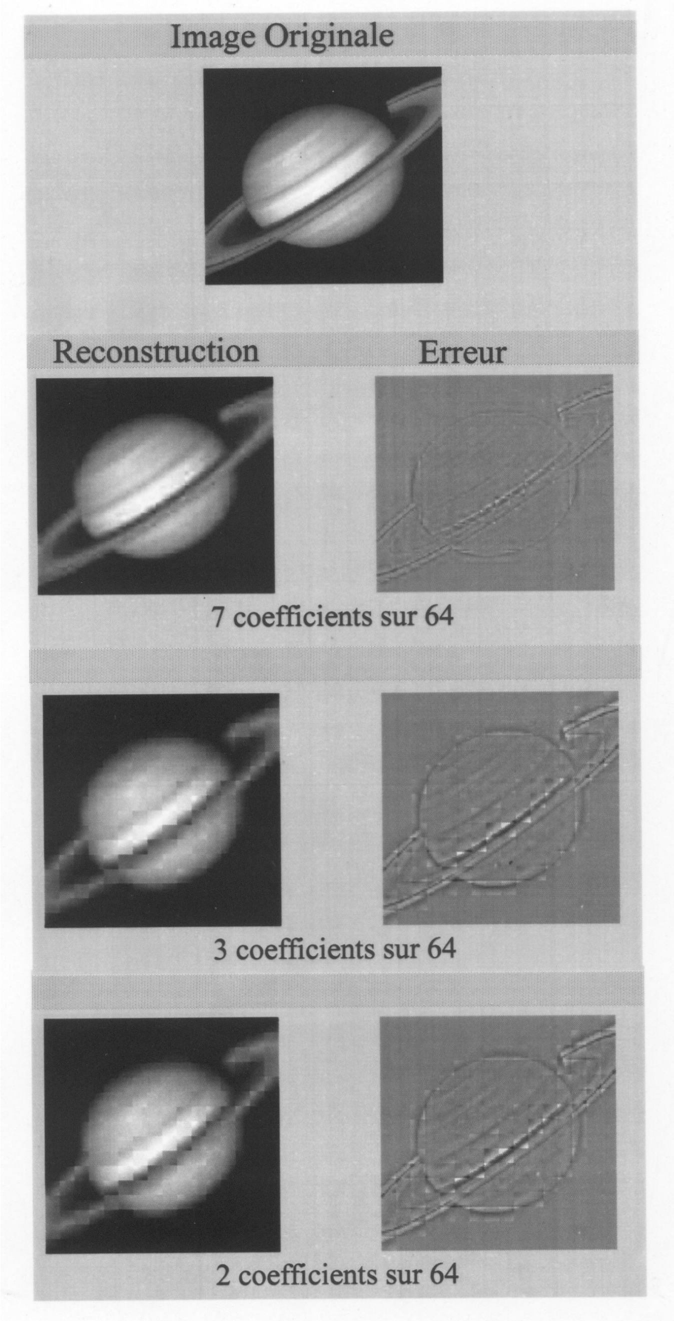
Pourcentage de l’erreur
en fonction du nombre de composantes
transmises sur les 16 originales
(signal théorique processus de Markov)
pour différentes transformées

Figure 5.19 Performance of different unitary transforms with respect to basis restriction errors (J_m) versus the number of basis (m) for a stationary Markov sequence with $N = 16$, $\rho = 0.95$.



Pourcentage de l'erreur
en fonction du taux de compression
pour une image 256*256
pour différentes transformées

III- Codage avec perte : méthodes par transformées



La transmission progressive d'images
est possible via la DCT

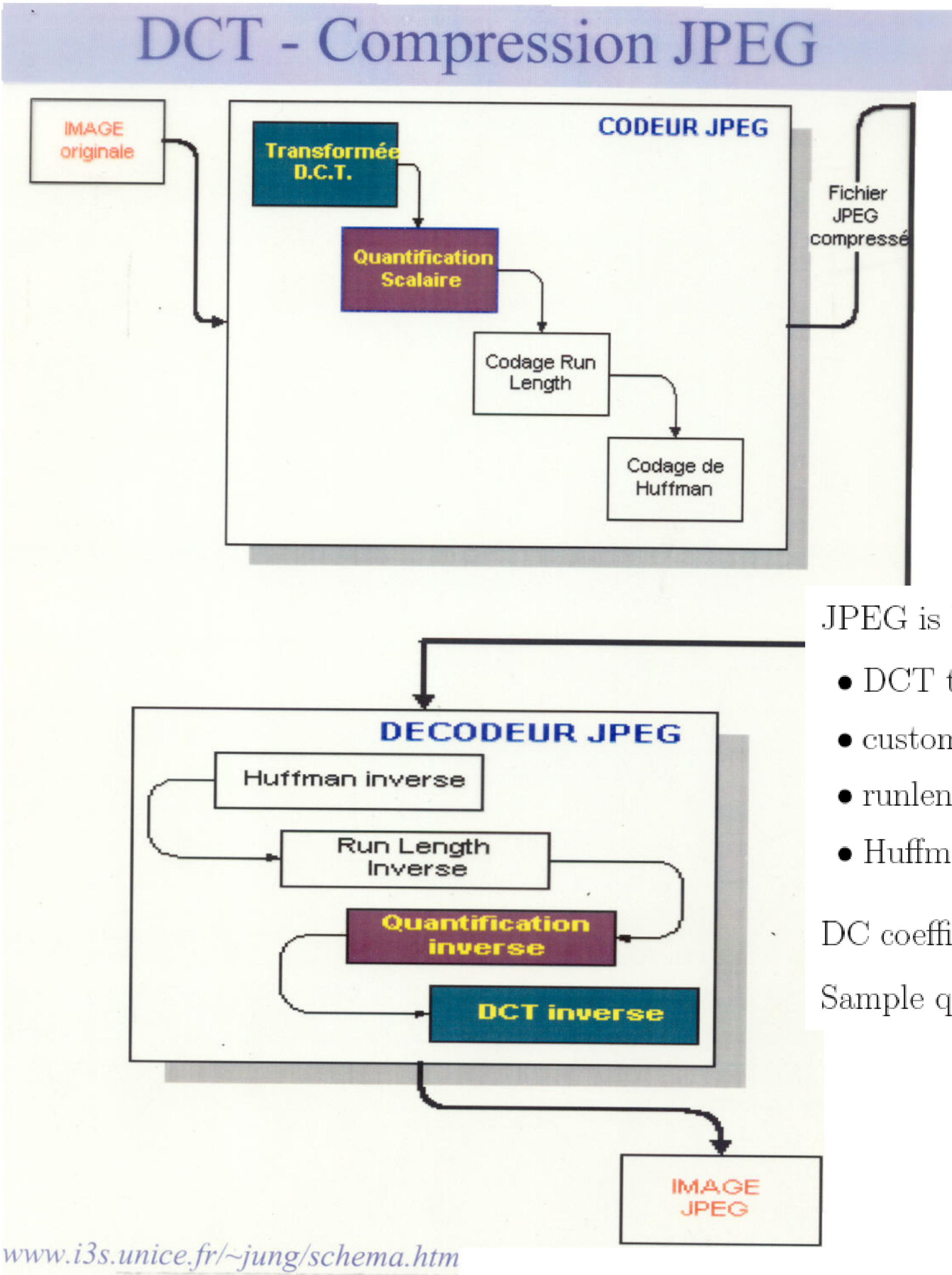
Why DCT so important?

- real
- can be computed by an FFT
- excellent energy compaction for highly correlated data
- good approximation to Karhunen-Loeve transform for Markov behavior and large images

Won in the open competition for an international standard.

- 1985: Joint Photographic Experts Group (JPEG) formed as an ISO/CCITT joint working group
- 6/87 12 proposals reduced to 3 in blind subjective assessment of picture quality
- 1/88 DCT-based algorithm wins
- 92-93 JPEG becomes intl standard

III- Codage avec perte : méthodes par transformées



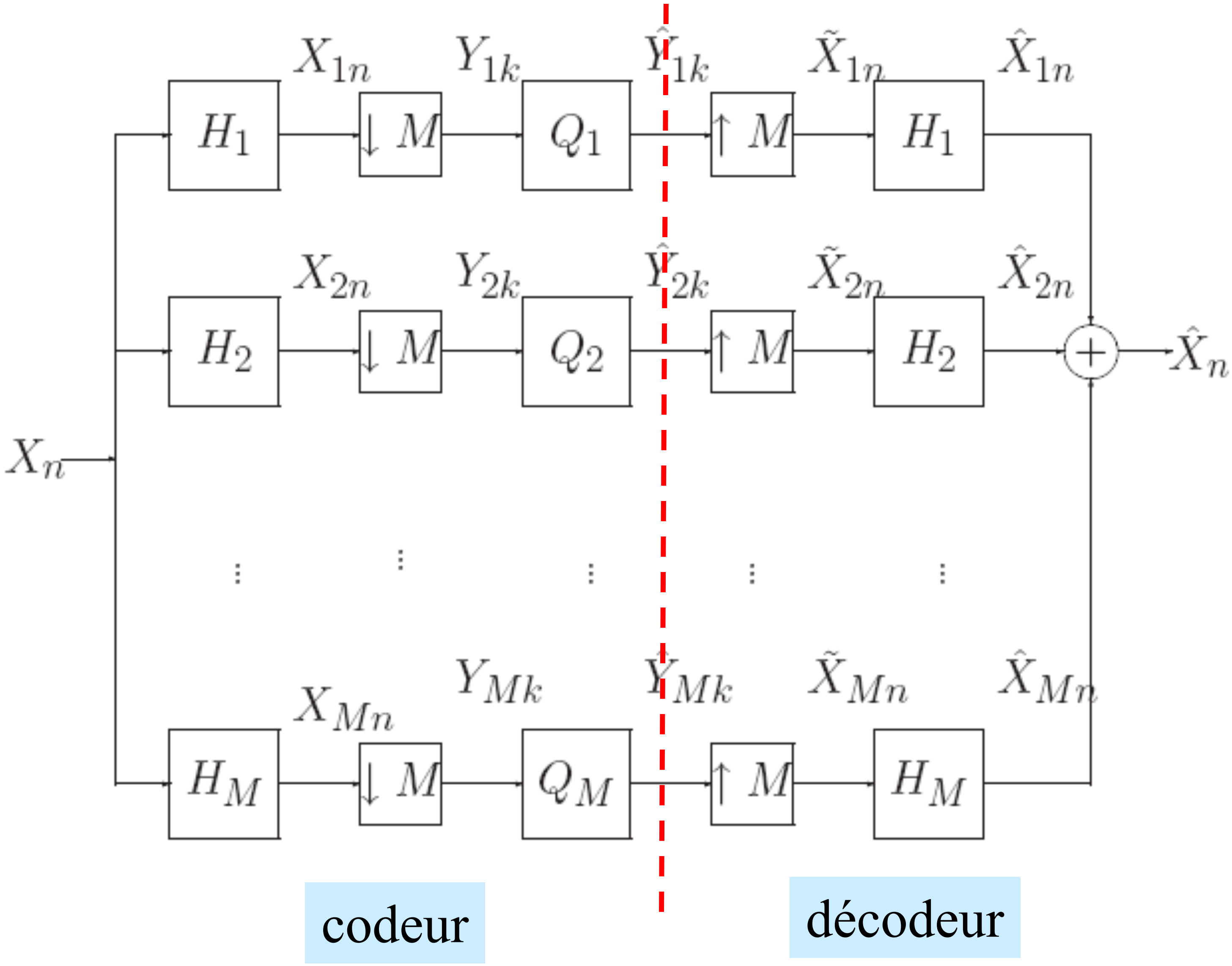
JPEG is basically

- DCT transform coding (DCT) on 8×8 pixel blocks
- custom uniform quantizers (user-specified Quantization tables)
- runlength coding
- Huffman or arithmetic coding on (runs, levels)

DC coefficient coded separately, DPCM

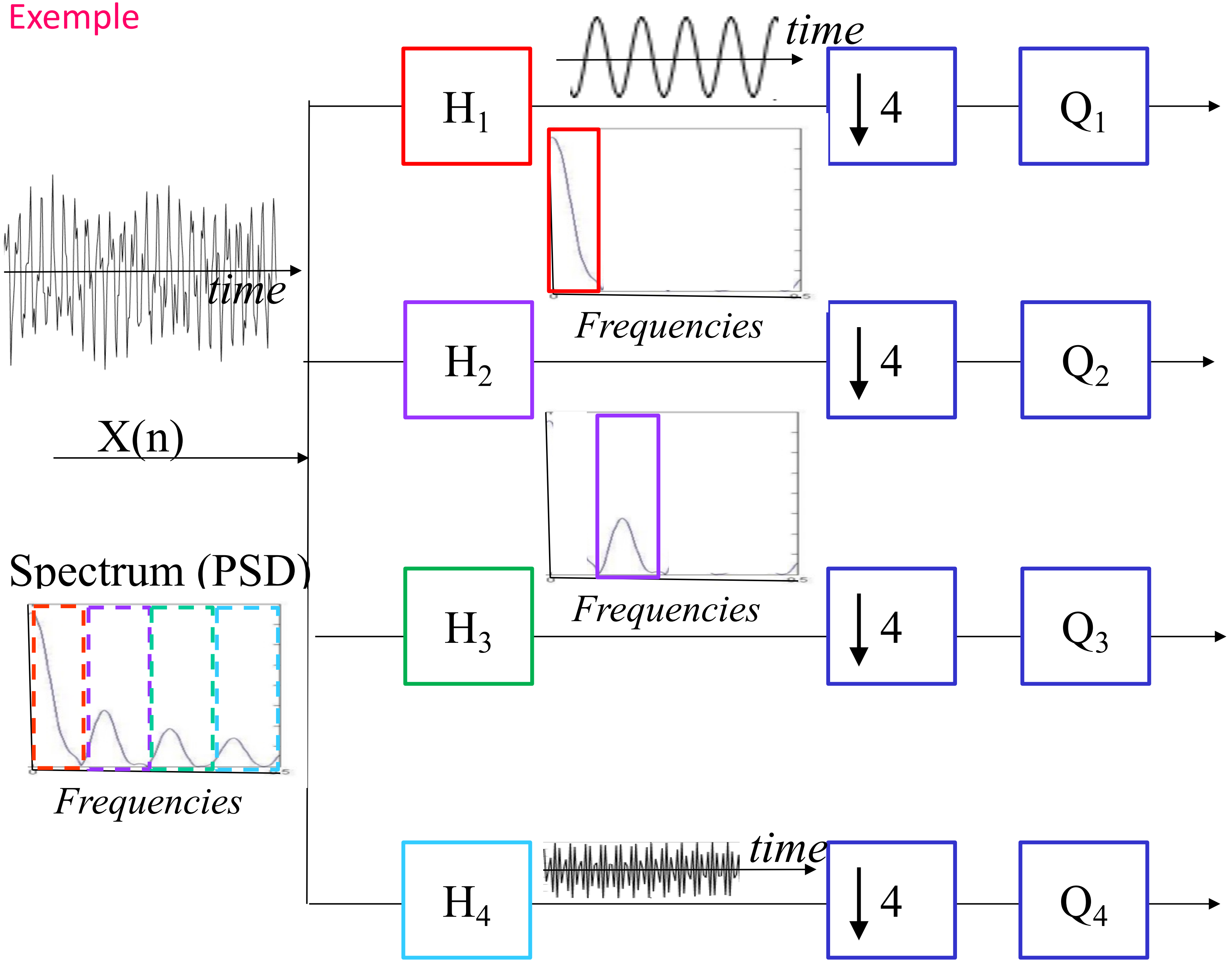
Sample quantization table

Une autre famille : le codage en sous-bande (par ondelettes)

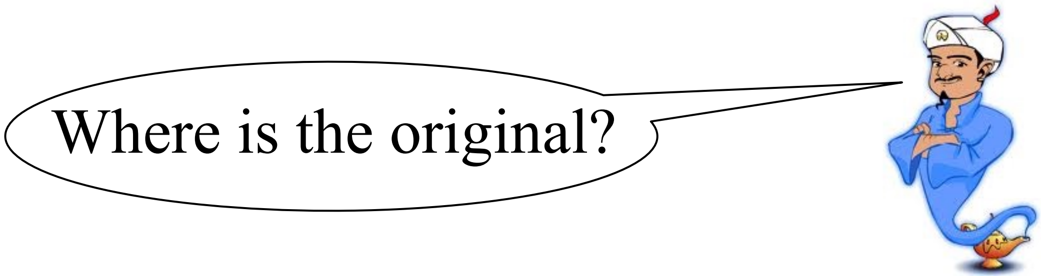


III- Codage avec perte : méthodes par transformées – codage en sous-bandes

Exemple



DWT + SPIHT example - PSNR = 37.12 dB, 0.5 bpp & Original: 8 bpp



Baseline JPEG: compressed 45:1

Wavelets & SPIHT: compressed 50:1

III- Codage avec perte : méthodes par transformées

Compression de données
Codage de source

I - Introduction

II- Codage sans perte

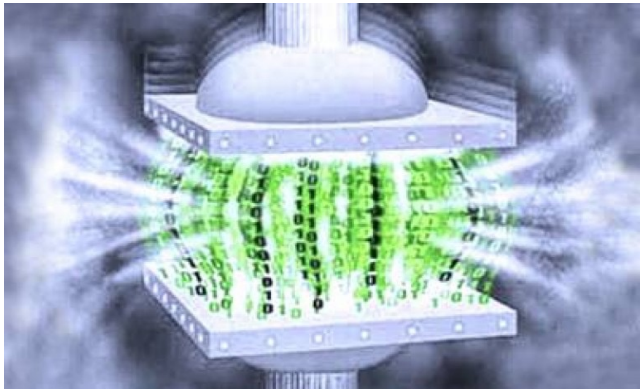
- les bases : la théorie de l'information
- codage d'Huffman

III- Codage avec perte : quantification scalaire,

IV- Codage avec perte : méthodes prédictives,

V- Codage avec perte : méthodes par transformées

TD



corinne.mailhes@n7.fr