M2.L4: Série d’exercices sur la compression de données

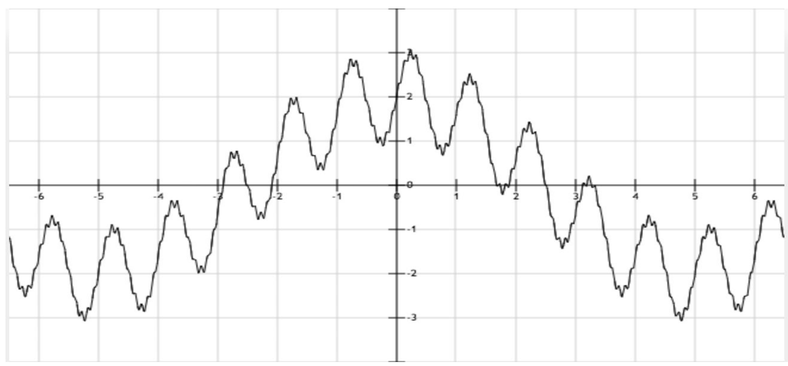
1 Filtrage et échantillonnage

1.1 Filtre Passe-Bande

Nous définissons le *filtre passe bande idéal* de **fréquence de coupure haute f_ch** et **fréquence de coupure basse f_cb** -- telles que f_ch > f_cb--, le filtre qui ne conserve que les signaux de fréquence compris entre f_cb et f_ch.

a) Comment le filtre passe-bande s’exprime-t’il en fonction d’un passe haut et d’un passe bas?

Soit le signal périodique f(x) suivant:



$f(x) = 2*\cos(0.1*2\pi*x) + \sin(1*2\pi*x) + 0.1*\sin(10*2\pi*x)$

b) Comment conserver seulement les 2 basses fréquences ? Esquisser le signal résultant.

c) Comment conserver seulement les 2 hautes fréquences ? Esquisser le signal résultant.

d) Comment conserver seulement la fréquence intermédiaire ? Esquisser le signal résultant.

1.2 Filtrer avant d’échantillonner

La question 1d) de la série M2.L1-L2 a précisé que lorsqu'on joue une note de musique sur instrument (p.ex. la note **La** à 440 Hz), on joue bien plus qu’une sinusoïde pure! Le son produit est en fait une somme de sinusoïdes dont les fréquences sont des multiples entiers de la fréquence *fondamentale* f_0 :

$$X(t) = \sum_{n \geq 1} a_n \sin(2\pi n f_0 t), \quad t \in \mathbb{R}$$

Les fréquences nf_0 sont les fréquences des *harmoniques* qui composent la note. Les amplitudes a_n des harmoniques dépendent de l’instrument sur lequel est joué la note ; elles déterminent le *timbre* de l’instrument (i.e. l’allure du signal X(t)).

a) Quelle est la bande passante du signal X?

b) Pour enregistrer cette note de musique, on échantillonne le signal X à une certaine fréquence **fe**. Pour éviter l'effet stroboscopique, il est nécessaire de filtrer le signal avant de l'échantillonner. Pour chacune des fréquences fondamentales **f0** et fréquences d'échantillonnage **fe** ci-dessous, déterminer quelle(s) fréquence(s) de coupure **fc** il est possible d'utiliser afin de préserver un nombre maximum d'harmoniques du signal tout en évitant l'effet stroboscopique; déterminer également le nombre d'harmoniques préservées dans chaque cas :

1. f0 = 440 Hz (La) et fe = 44.1 kHz
2. f0 = 495 Hz (Si) et fe = 44.1 kHz
3. f0 = 330 Hz (Mi) et fe = 8’820 Hz

c) Beaucoup de systèmes de téléphonie mobile filtrent les signaux transmis à 3.4 kHz (ce qui permet de transmettre à peu près correctement la voix humaine): combien de bits au minimum sont nécessaires pour enregistrer correctement une conversation de 5 minutes avec un tel système (avec une représentation des nombres réels en virgule flottante sur 64 bits)?

2 Algorithme de Shannon-Fano

a) En utilisant l’algorithme de Shannon-Fano, représentez la séquence suivante (sans tenir compte des espaces) par une séquence de bits:

INFORMATION CALCUL ET COMMUNICATION

b) Combien de bits par lettre en moyenne sont-ils nécessaires pour représenter cette séquence?

c) Calculez l’entropie de la séquence. Vérifie-t-on les inégalités vues au cours?

Indication: Si le calcul de l’entropie vous fatigue, essayez http://www.shannonentropy.netmark.pl (attention à supprimer les espaces! De plus il y a un risque de récupérer des caractères non-imprimables si vous faites du copier-coller, ce qui produit un résultat faux)

d) Si vous vous restreignez à utiliser un code qui ne tient pas compte des probabilités d’apparition et qui utilise exactement le même nombre de bits pour chaque lettre, de combien de bits aurez-vous besoin pour représenter la séquence?

Voici une autre séquence de lettres (où on oublie à nouveau les espaces, les accents, les traits d’union, les virgules et les apostrophes!):

DIDON DINA, DIT-ON, DU DOS D’UN DODU DINDON

e) A priori, pouvez-vous deviner laquelle des deux séquences ci-dessus a la plus faible entropie?

f) Représentez à nouveau la séquence par une séquence de bits en utilisant l’algorithme de Shannon-Fano.

g) Répondez à nouveau aux questions c) et d) pour cette deuxième séquence de lettres.

3 Codage par plages (run-length encoding)

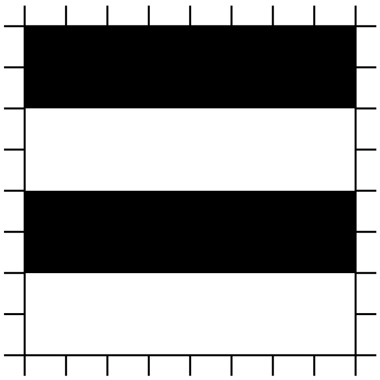


Image 1

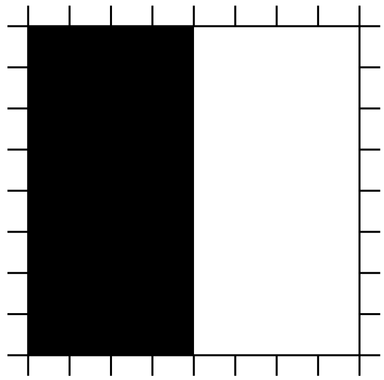


Image 2

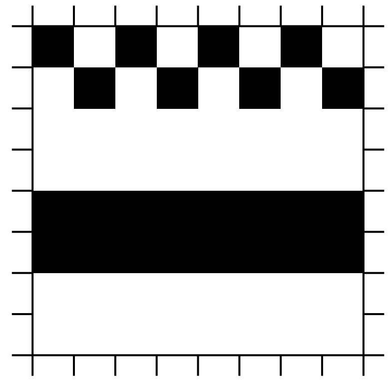


Image 3

Dans cet exercice, on considère un autre type de compression, utilisé principalement pour les images en noir et blanc comme celles présentées ci-dessus. Il s'agit du *codage par plages* ou run length encoding (RLE) en anglais. L'idée est la suivante: dans une image en noir et blanc, chaque pixel est représenté par un 1 (noir) ou un 0 (blanc). Pour comprimer une image avec 8 × 8 = 64 pixels, on transforme tout d'abord celle-ci un une séquence de 64 bits, en "lisant" l'image ligne par ligne. Ainsi l'image 1 ci-dessus est représentée de manière "brute" par la séquence de bits:

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

Vu le grand nombre de 1 et de 0 consécutifs dans cette séquence, il semble qu'on peut économiser de l'espace-mémoire en procédant comme suit: on divise la séquence en paquets de 4 bits de longueur; dans chaque paquet, le premier bit symbolise la couleur (0 ou 1) de la suite de pixels et les 3 bits suivants indiquent en binaire le nombre de pixels consécutifs de cette couleur, moins 1. Exemples:

0010 signifie "3 pixels consécutifs de couleur 0"
1101 signifie "6 pixels consécutifs de couleur 1"

a) Quel est le codage RLE des trois images ci-dessus, et quelle est la longueur de ce code pour chaque image (i.e. combien épargne-t-on de bits par rapport à la taille originale de l'image qui est de 64 bits)?

b) Si vous avez résolu correctement la partie a), vous aurez constaté que le codage RLE de la seconde image est bien plus long que celui de la première image, alors que ces images sont très similaires par nature. Proposez une modification du format du codage qui considère qu'un nombre fixe de bits au début du codage sert à définir une convention utilisée pour le codage de toute l'image (en conservant le même type de paquet de 4 bits).

c) Si vous avez résolu correctement la partie a), vous aurez également constaté que le codage RLE de la troisième image pose problème. Cette fois-ci nous vous proposons de modifier la taille de paquet (sachant que cette taille reste utilisée pour toute l'image) et le rôle d'un ou de plusieurs bits de chaque paquet. Le but est d'introduire une plus grande flexibilité dans le codage pour pouvoir aussi bien coder des plages uniformes et des zones arbitrairement variées (image 3 ou autres possibilités). Que proposez vous?

4 Algorithme de Huffman (il y a beaucoup à lire dans cet exercice mais pas beaucoup à faire !)

Nous nous intéressons dans cet exercice au code de Huffman qui est optimal (pour une compression *sans perte*) : on ne peut pas faire plus court !

4.1 Rappel de l'algorithme de Huffman

On part aussi du tableau des lettres et de leur nombre d'apparition (ou probabilité).
L'algorithme procède alors itérativement comme suit :

1. Trouver les 2 lettres les moins fréquentes et les regrouper (cela définit une question). En cas d'égalité, en choisir 2 au hasard (ça ne change pas la longueur moyenne du code). Par exemple, si les 2 lettres les moins fréquentes sont A et F, la question serait (par exemple) est-ce A ?

2. Dans le tableau des nombres d'apparitions des lettres, supprimer les 2 dernières lettres considérées et les regrouper comme une seule nouvelle lettre, avec comme nombre d'apparitions la somme des deux nombres. Pour continuer sur l'exemple précédent, si A apparaissait 1 fois et F 2 fois, alors on supprime A et F du tableau et on introduit la nouvelle lettre A|F (qui veut dire A ou F) avec comme nombre d'apparitions 3.

3. Recommencer en 1 tant qu'il y a des lettres .

Une fois l'arbre des questions construit, procéder comme pour le code de Shannon-Fano en décidant d'affecter le symbole 0 à toutes les réponses non et le symbole 1 à toutes les réponses oui .

4.2 Exemple

Considérons les lettres A,B,C,D et E avec les probabilités respectives de 1/3, 1/8, 1/8, 1/4 et 1/6 ; par exemple dans (sans les espaces) :

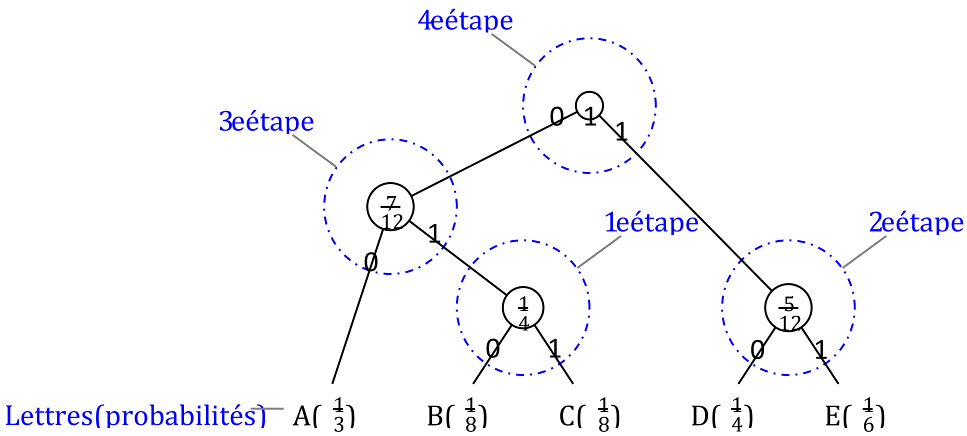
A A AD AD ADE ABCDE ABCDE ABCDE

On commence donc avec le tableau

	A	B	C	D	E
probabilité	$\frac{1}{3}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{6}$

On regroupe donc B et C (les deux moins probables) sous une question est-ce C ? . Puis on les regroupe dans le tableau pour recommencer. On recommence donc avec les 2 lettres les moins probables, donc E (1/6) et, par exemple, D (1/4). A la place de D, on aurait aussi pu choisir ici la nouvelle lettre B—C qui a aussi une probabilité de 1/4. Il existe plusieurs codes de Huffman équivalents pour une même séquence. Il suffit d'en choisir un (et bien sûr d'en convenir avec son destinataire, comme pour tout code !).

Et on continue ainsi de suite. L'arbre des questions correspondant est donné dans la figure suivante, où au lieu d'écrire les questions elles-mêmes nous avons représenté les probabilités sommées de toutes les alternatives couvertes par chaque question.



Exemple de code de Huffman.

	A	B—C	D	E
probabilité	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{6}$

Les mots de codes correspondants sont (pour A, B, C, D et E respectivement) : 00, 010, 011, 10 et 11.

4.3 Mise en pratique

Supposons que nous ayons un texte avec les nombres d'apparitions suivants :

	A	B	C	D	E
nb. app.	39	17	16	15	13

1. Construire le code de Shannon-Fano correspondant.
2. Construire le code de Huffman correspondant.
3. Quelle est la longueur moyenne de chaque code ? Comparer à la limite théorique donnée par le théorème de Shannon.