M2.L4: Série d'exercices sur la compression de données [Solutions]

1 Filtrage et échantillonnage

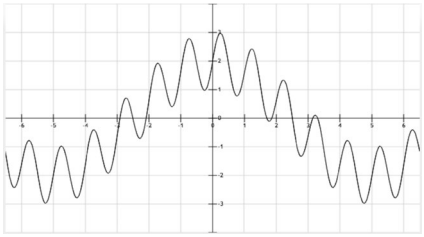
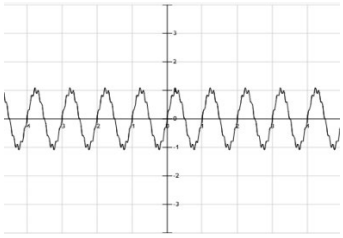
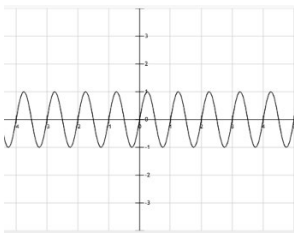
1.1 Filtre Passe-Bande

1. Il s'agit de la composée d'un filtre passe bas de fréquence f_ch et d'un passe haut de fréquence f_cb

2. Comment conserver seulement les 2 basses fréquences ? Utiliser un passe-bas de fréquence 5 Hz

3. Comment conserver seulement les 2 hautes fréquences ? Utiliser un passe-haut de fréquence 0.5 Hz

4. Comment conserver seulement la fréquence intermédiaire ? Utiliser un passe-bande 0.5 Hz - 5 Hz

signal avec les 2 basses fréquences ; avec les 2 hautes fréquences ; avec la fréquence intermédiaire

1.2 Filtrer avant d'échantillonner

a) La bande passante du signal est infinie, car celui-ci possède un nombre infini de fréquences nf0 allant jusqu'à l'infini et toutes les amplitudes a_n des harmoniques sont supposées strictement positives ici. Bien sûr, la valeur de a_n décroît rapidement avec n; les harmoniques correspondant à de grandes valeurs de n sont donc à peine audibles en pratique (aussi parce que notre oreille ne perçoit simplement pas les fréquences au-delà de 22 kHz).

b) Pour assurer que la condition d'échantillonnage soit satisfaite, il faut que fe > 2B, où B est la bande passante du signal après filtrage.

1. Si f0 = 440 Hz et fe = 44.1 kHz, il faut que B < 22.05 kHz: en filtrant le signal avec une fréquence de coupure fc comprise entre 22 kHz (50x 440Hz) et 22'049 Hz (fe/2 – 1Hz), on s'assure de préserver les 50 premières harmoniques jusqu'à f_{50} = 22 kHz, tout en évitant l'effet stroboscopique lors de la reconstruction. Remarquez qu'il n'y a pas besoin que fe > 2fc, car le signal ne contient aucune composante entre 22 kHz et 22'439 Hz (= 51x 440Hz – 1Hz); on peut donc filtrer celui-ci avec n'importe quelle fréquence de coupure fc dans cet intervalle.

2. Avec le même raisonnement, on conclut qu'avec une fréquence de coupure fc comprise entre 21'780 Hz et 22'274 Hz, on préserve les 44 premières harmoniques du signal.

3. Encore avec le même raisonnement, on conclut qu'avec une fréquence de coupure fc comprise entre 4'290 Hz et 4'619 Hz, on préserve les 13 premières harmoniques du signal.

c) Pour échantillonner correctement le signal, il faut utiliser une fréquence d'échantillonnage juste au-dessus de 6'800 Hz (disons égale à ce chiffre pour simplifier), donc le nombre de bits à enregistrer est de 300 (secondes)* 6'800 * 64 = 130'560'000 bits soit environ 131 Mégabits.

Algorithme de Shannon-Fano

a) En utilisant le jeu des questions, voici le code qu'on trouve:

lettre	nb apparitions	nb questions	mot de code
I	4	3	111
N	4	3	110
O	4	3	101
C	4	3	100
M	3	3	011
A	3	4	0101
T	3	4	0100
L	2	4	0011
U	2	4	0010
F	1	4	0001
R	1	5	00001
E	1	5	00000

b) Au total, on a donc besoin de 19 × 3 + 11 × 4 + 2 × 5 = 111 bits, i.e. L(C)= 111/32= 3.46 bits par lettre.

c) L'entropie de la séquence est égale à $H(X) = 3.429$. On vérifie donc bien que $H(X) \leq L(C) \leq H(X) + 1$, on voit en fait que $L(C)$ est beaucoup plus proche de $H(X)$ que de $H(X) + 1$.

d) La séquence contient 12 lettres différentes: si on veut utiliser le même nombre de bits par lettre, on a donc besoin de 4 bits par lettre au minimum (car $2^4 = 16$, qui est la puissance de 2 la plus proche au-dessus de 12), et donc de 32 × 4 = 128 bits au total. On utilise donc 17 bits de plus qu'avec le code de Shannon-Fano.

e) La séquence commençant par DIDON... a un très grand nombre de *D* (11 pour être plus précis): on s'attend donc à une plus faible entropie.

f) En utilisant à nouveau le jeu des questions, voici le code qu'on trouve:

lettre	nb apparitions	nb questions	mot de code
D	11	2	11
N	6	2	10
O	5	3	011
I	4	3	010
U	3	3	001
A	1	4	0001
T	1	5	00001
S	1	5	00000

f) Le nombre total de bits utilisés est cette fois-ci de 17 × 2 + 12 × 3 + 1 × 4 + 2 × 5 = 84 bits, i.e. L(C)= 2.625 bits par lettre. Quand à l'entropie de la séquence, elle est égale à $H(X) = 2.564$. A nouveau, on vérifie bien que $H(X) \leq L(C) \leq H(X) + 1$, et que $L(C)$ est plus proche de $H(X)$ que de $H(X) + 1$, même si les probabilités des lettres sont assez irrégulières dans le cas présent.

g) La séquence contient 8 lettres différentes: si on veut utiliser le même nombre de bits par lettre, on a donc besoin de 3 bits par lettre au minimum (car $2^3 = 8$) et donc de 32×3 = 96 bits au total. On utilise donc 12 bits de plus qu'avec le code de Shannon-Fano.

3 Codage par plages (run-length encoding)

a) Vu que chaque ligne de la première image est unicolore, son code RLE est donné par

1111|1111|0111|0111|1111|1111|0111|0111 (longueur totale de 32 bits)

Sur chaque ligne de la deuxième image, on voit le même motif de 4 pixels noirs (encodés par 1011) suivis de 4 pixels blancs (encodés par 0011), donc le code RLE de l'image est

1011|0011|1011|0011|... (longueur totale de 64 bits)

Sur la troisième image, les deux premières lignes sont des séries alternées de noir et blanc, donc avec le codage RLE, chacun des pixels est représenté par 4 bits! (0000 si le pixel est blanc, ou 1000 si le pixel est noir). Donc rien que l'encodage des deux premières lignes requiert 16 × 4 = 64 bits ! (On néglige ici les deux pixels blancs qui se suivent de la fin de la première ligne au début de la seconde). Les six prochaines lignes ne requièrent chacune que 4 bits, comme sur la première image. Au total, on a donc besoin de 64 + 24 = 88 bits; clairement pas une compression!

b) Dans cet exercice, on a choisi d'encoder l'image ligne par ligne, mais rien ne nous empêche de faire la même chose colonne par colonne. Pour préciser notre choix dans le code RLE, il suffit d'ajouter un bit au début: 0 pour "ligne par ligne" et 1 pour "colonne par colonne". On peut ainsi représenter la deuxième image avec 33 bits au total:

1|1111|1111|1111|1111|0111|0111|0111|0111

c) On voit bien que le codage RLE des deux premières lignes est loin d'être optimal. Une façon de remédier à ça est de changer la structure des paquets, en ajoutant un bit au début de chaque paquet (dont la longueur passe donc à 5 bits): si ce bit vaut 0, alors les quatre prochains bits sont à interpréter au sens du code RLE ci-dessus; par contre, si le premier bit du paquet vaut 1, alors les quatre prochains bits sont simplement les couleurs des 4 pixels de l'image. Ainsi, la troisième image sera encodée par la séquence de bits suivante:

11010|11010|10101|10101|00111|00111|01111|01111|00111|00111

pour une longueur totale de 50 bits (51 si on ajoute le premier bit 0 de la partie b) pour dire qu'on procède ligne par ligne).

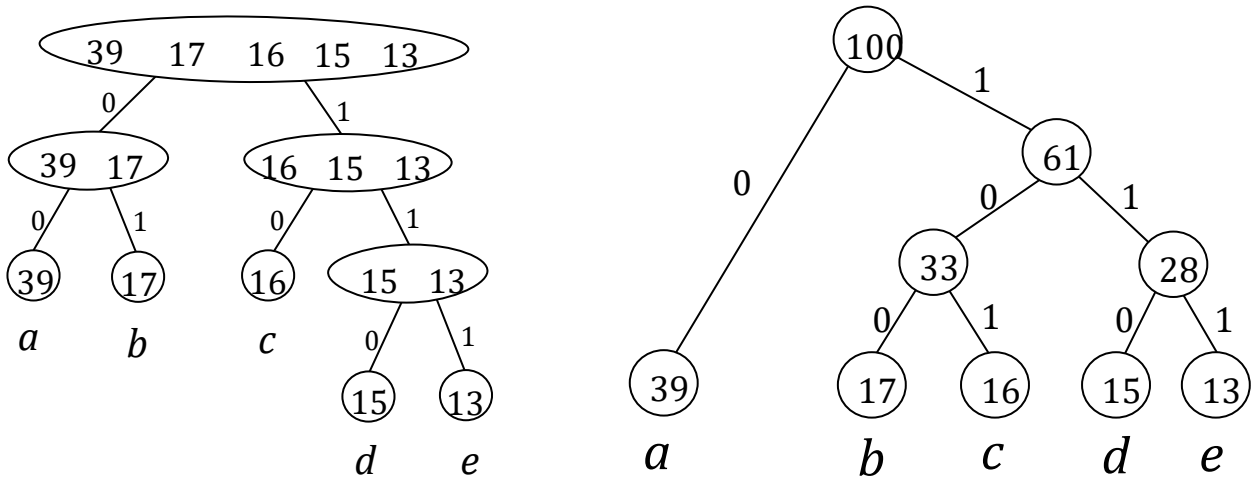


Figure 1: à gauche: code de Shannon-Fano code; à droite: code de Huffman

4 Algorithme de Huffman

L'arbre binaire pour les deux différents codes apparaît sur la figure ci-dessus. Le dictionnaire obtenu par le code de Shannon-Fano est:

$$a \rightarrow 00,\ b \rightarrow 01,\ c \rightarrow 10,\ d \rightarrow 110,\ e \rightarrow 111$$

Donc le nombre total de bits utilisé par le code de Shannon-Fano est: (39 + 17 +16)*2 + (15 + 13)*3 = 228.
Le nombre moyen de bits est 228/100 = 2,28 bits/lettre

Le dictionnaire obtenu par le code de Hufman est:

$$a \rightarrow 0,\ b \rightarrow 100,\ c \rightarrow 101,\ d \rightarrow 110,\ e \rightarrow 111$$

Donc le nombre total de bits utilisé par le code de Huffman est: 39*1 + (17 + 16 + 15 + 13)*3 = 222.
Le nombre moyen de bits est 222/100 = 2,22 bits/lettre

La limite théorique donnée par le théorème de Shannon est l'entropie:

Entropie = $-0.39\log_2(0.39) - 0.17\log_2(0.17) - 0.16\log_2(0.16) - 0.15\log_2(0.15) - 0.13\log_2(0.13) \approx 2.18$ bits/lettre

A noter que dans ce cas, la borne inférieure ne peut pas être atteinte. Du fait que les codes de Huffman sont optimaux (pour la compression sans pertes), on ne peut donc pas trouver dans ce cas un schéma de compression qui fasse mieux que 2.22 bits/lettre. Le théorème de Shannon est vrai pour *tous* les codes. Dans certains cas, la borne inférieure que constitue l'entropie ne peut être atteinte (de fait, le théorème ne dit pas qu'elle doit l'être).