TP10 – Manipulation de signaux audionumériques

Transformation de Fourier à court terme d'un signal audio

La première *représentation temps-fréquence* d'un signal audio a été proposée en 1946 par Dennis Gabor, physicien hongrois ayant obtenu le prix Nobel en 1971 pour l'invention de l'holographie. La *transformation de Gabor* consiste à multiplier le signal par une fenêtre glissante afin d'obtenir le « spectre instantané » du signal. Bien que Gabor ait utilisé une gaussienne comme fenêtre glissante, il est possible d'utiliser n'importe quelle fenêtre. Cette généralisation s'appelle la *transformation de Fourier à court terme* (TFCT, pour *Short Time Fourier Transform*). La transformée de Fourier à court terme Y d'un signal unidimensionnel *continu* y s'écrit :

$$Y(\tau, f) \coloneqq \mathbf{TFCT}\,\{y\}\,(\tau, f) = \mathbf{TF}\{y\, w_\tau\}(f) \tag{1}$$

où w_τ désigne la fenêtre glissante, qui peut être positionnée à un instant τ variable. Par conséquent, alors que le signal d'origine y dépend d'une seule variable réelle (le temps t), sa transformée de Fourier à court terme Y, qui est une fonction complexe, dépend de deux variables réelles (la position temporelle τ de la fenêtre glissante et la fréquence f), obtenue par concaténation de spectres instantanés.

Dans le cas de signaux discrets $\mathbf{y} \in \mathbb{R}^L$, échantillonnés à la fréquence f_{ech}, en choisissant une fenêtre $\mathbf{w} \in \mathbb{R}^N$ contenant N échantillons, et un décalage H entre deux positions successives de la fenêtre, nous obtenons :

$$\mathbf{Y}(m, k) \coloneqq \mathbf{TFCT}\,\{\mathbf{y}\}\,(m, k) = \mathbf{TFD}\{\mathbf{y}\, \mathbf{w}_{mH}\}(k) \tag{2}$$

où $k \in \{0, \ldots, N-1\}$ et $m \in \{0, \ldots, \lfloor \frac{L-N}{H} \rfloor + 1\}$ indexent, respectivement, les fréquences et les instants des coefficients du sonagramme. Les formules suivantes permettent de convertir ces indices en Hz et en s :

$$f_{\text{coef}}(k) \coloneqq \frac{k \cdot f_{\text{ech}}}{N} \qquad\qquad t_{\text{coef}}(m) \coloneqq \frac{m \cdot H}{f_{\text{ech}}} \tag{3}$$

Les signaux manipulés étant réels, on peut conserver uniquement les coefficients de Fourier correspondant aux fréquences positives, c'est-à-dire à $k \in \{0, \ldots, N/2\}$, auquel cas le nombre de lignes de $\mathbf{Y}$ est égal à $N/2+1$. Quant au nombre de colonnes, il est égal au nombre de positions de la fenêtre glissante, qui dépend de la valeur H du décalage.

La représentation standard d'une transformée de Fourier consiste à afficher le *spectre de puissance* en utilisant une échelle en *décibels* (dB) :

$$S(\tau, f) \coloneqq |Y(\tau, f)|^2 \quad \Longrightarrow \quad S_{dB}(\tau, f) \coloneqq 10 \times \log_{10} |Y(\tau, f)|^2 \tag{4}$$

où la notation $|\cdot|$ désigne le module complexe. Une telle représentation est appelée *spectrogramme* de manière générale, et *sonagramme* dans le cas de données audio (cf. figure 1).

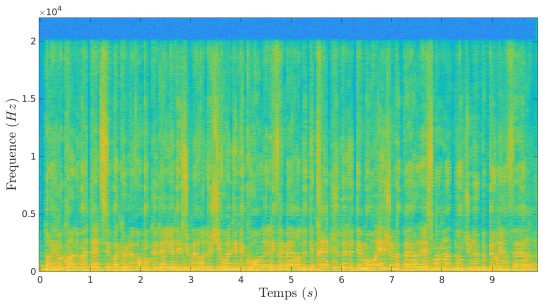

FIGURE 1 – Sonagramme d'un extrait audio (représentation en fausses couleurs).

Exercice 1 : calcul de la transformée de Fourier à court terme

Écrivez la fonction `TFCT`, appelée par le script `exercice_1`, permettant de calculer la transformée de Fourier à court terme du signal passé en paramètre. Observez l'influence des différents paramètres du script `exercice_1`, en particulier du type de fenêtre utilisée (fenêtre rectangulaire ou fenêtre de Hann).

Remarques importantes :

- Il est conseillé de découper le signal à l'aide de la fonction `buffer` de Matlab.
- Les colonnes de la matrice `TFCT` étant calculées à l'aide de la fonction `fft` de Matlab, les lignes de cette matrice correspondent aux fréquences, mais sont rangées dans l'ordre suivant : $f = 0, \ldots, f_{\max}$ pour la première moitié, $f = -f_{\max}, \ldots, 0$ pour la deuxième moitié. Il est demandé de ne garder que la partie correspondant aux fréquences positives (les autres informations étant redondantes si le signal est réel).
- Par défaut (ou en spécifiant `axis ij`), les axes des graphiques affichés par Matlab sont orientés vers la droite pour les abscisses, mais vers le bas pour les ordonnées. La commande `axis xy` permet de forcer l'axe des ordonnées à être orienté vers le haut.
- Dans la mesure où la fonction `fft` de Matlab, appliquée à une matrice, calcule la TFD colonne par colonne, il est conseillé d'écrire la fonction `TFCT` sans boucle `for`.

La transformation de Fourier étant inversible, il doit être possible de restituer le signal audio d'origine, sans aucune distorsion, à partir de la transformée de Fourier à court terme. En revanche, cela n'est plus le cas si l'on dispose uniquement de sa partie réelle (perte de la partie imaginaire) ou de son module complexe, c'est-à-dire du sonagramme (perte de la phase). Modifiez la fin du script `exercice_1` afin de comparer le son restitué dans ces deux cas de figure.

Exercice 2 : filtrage passe-bas, filtrage passe-haut et compression

La représentation *temps-fréquence* permet de réaliser des effets audio de manière très simple. Tout d'abord, il est possible de tronquer le sonagramme pour ne garder que certaines bandes de fréquences (basses ou hautes). De plus, un sonagramme indique quelles fréquences contribuent le plus à la reconstruction du signal à un instant donné. Il est donc possible de conserver uniquement une faible proportion des coefficients de Fourier les plus élevés, afin de réaliser une compression « naïve » du signal (la partie facultative du TP vise à effectuer une compression plus élaborée d'un signal audio, cf. exercice 4).

Le script `exercice_2` permet d'appliquer des modifications à un sonagramme. Écrivez la fonction `passe_bas` (resp. `passe_haut`) prenant en paramètre un sonagramme, l'échelle fréquentielle ainsi qu'une fréquence de coupure et permettant de mettre à zéro les coefficients situés au dessus (resp. au-dessous) de la fréquence de coupure. Enfin, écrivez la fonction `compression` visant à ne conserver, dans un sonagramme, que les m coefficients de Fourier les plus élevés (en module) de chaque colonne.

Testez différents jeux de paramètres pour ces trois effets et écoutez les résultats produits. Vous constaterez qu'une forte compression permet, dans une certaine mesure, de restituer correctement le signal d'origine.

Exercice 3 : étirement temporel et transposition

Afin d'étirer temporellement un signal audio, une première idée consite à changer la vitesse de lecture, ou de façon équivalente, de faire comme si le signal avait été échantillonné à une fréquence plus élevée que sa véritable fréquence d'échantillonnage. Cependant, cette méthode altère également le contenu fréquentiel du signal.

Pour résoudre ce problème, un algorithme d'*étirement temporel* (cf. Algorithme 1) traite le module du sonagramme du signal comme une image, que l'on peut facilement étirer horizontalement par simple interpolation linéaire. Le problème réside plutôt dans la manipulation de la phase. Cet algorithme propose d'utiliser, comme différence de phase entre deux colonnes du sonagramme interpolé, la différence de phase entre les deux colonnes du sonagramme d'origine ayant servi à l'interpolation.

Algorithme 1 – Étirement temporel

$\mathbf{Y} \leftarrow \mathbf{TFCT}\{\mathbf{y}\}$
$\mathbf{C} \leftarrow 1 : \text{pourcentage} : \text{size}(\mathbf{Y}, 2)$
$\phi \leftarrow \text{angle}(\mathbf{Y}(:, 1))$
for $i \in \{1, \ldots, \text{length}(\mathbf{C})\}$ **do**
  $c \leftarrow \text{floor}(\mathbf{C}(i))$
  $\alpha \leftarrow \mathbf{C}(i) - c$
  $\rho \leftarrow (1 - \alpha)\, \mathbf{Y}(:, c) + \alpha\, \mathbf{Y}(:, c + 1)$
  $\mathbf{Y}'(:, c) \leftarrow \rho\, e^{j\phi}$
  $d\phi \leftarrow \text{angle}(\mathbf{Y}(:, c + 1)) - \text{angle}(\mathbf{Y}(:, c))$
  $\phi \leftarrow \phi + d\phi$
end for

Le script `exercice_3` permet d'appliquer un étirement temporel et/ou une transposition à un signal. Écrivez la fonction `etirement_temporel`, qui ayant reçu en entrée un signal, sa fréquence d'échantillonnage et un pourcentage d'accélération, génère un nouveau signal accéléré, mais de même contenu fréquentiel que celui du signal passé en paramètre.

Application au *sampling*

La technique du *sampling* consiste à récupérer des extraits sonores issus d'enregistrements existants pour les réutiliser dans un nouveau contexte musical. Elle s'apparente à la technique du « collage » en arts plastiques. Les extraits peuvent être de nature et de longueur très variées : il peut s'agir d'un motif rythmique ou mélodique entier, ou bien de quelques notes soigneusement sélectionnées. De nombreux effets peuvent aussi être ajoutés.

Le répertoire `ExSamples` contient quelques partitions de morceaux composés à partir de *samples*. Le script `app_sampling` permet de reconstruire le rendu final de chacun de ces exemples, et de le coupler à un affichage animé. Lancez ce script, en chargeant les différents exemples contenus dans le répertoire `ExSamples`.

Exercice 4 : compression avancée (facultatif)

La fonction `compression` de l'exercice 2 ne conserve, dans chaque colonne du sonagramme, que les coefficients qui contribuent le plus à la reconstitution du signal. Calculez le taux de compression réalisé par cette fonction. Pour des signaux de parole, il est possible d'obtenir une reconstruction intelligible du signal avec un taux de compression élevé (les résultats sont plus décevants avec les signaux musicaux).

Proposez une stratégie de décimation du sonagramme permettant d'améliorer le taux de compression de la méthode de compression de l'exercice 2.