

TP11 – Transformation de Gabor et compression audio

La première représentation temps-fréquence d'un signal temporel est due à Gabor, physicien hongrois ayant obtenu le prix Nobel en 1971 pour l'invention de l'holographie. La *transformation de Gabor* (1946), qu'on appelle également *transformation de Fourier locale* ou *transformation de Fourier à fenêtre glissante*, est bien antérieure à la transformation en ondelettes (1984). Elle consiste à multiplier le signal par une *fenêtre glissante*, afin d'obtenir l'*analyse fréquentielle instantanée* d'un signal.

Bien que Gabor ait utilisé une gaussienne comme fenêtre glissante, c'est le plus souvent une « fonction créneau » qui est utilisée. La transformée de Gabor $\mathcal{G}\{s(t)\}$ d'un signal temporel $s(t)$ s'écrit alors :

$$\mathcal{G}\{s(t)\}_{(\tau,f)} = \mathcal{F}\{s(t) w(t - \tau)\}_{(f)}$$

où \mathcal{F} désigne la transformée de Fourier et $w(t - \tau)$ la fenêtre glissante, qui peut être positionnée à un instant τ variable. Par conséquent, cette transformée est une fonction du temps τ et de la fréquence f (pour un signal temporel, on préfère généralement utiliser la fréquence, exprimée en Hertz, notés Hz , plutôt que la pulsation). La représentation d'une transformée de Gabor se fait dans le plan (temps en abscisse, fréquence en ordonnée), sous la forme d'une pseudo-image dont le niveau de gris est proportionnel à la partie réelle. Une telle pseudo-image s'appelle un *sonagramme* (ou *spectrogramme sonore*) : cf. l'exemple de la figure 1, où la carte des couleurs est la carte des couleurs par défaut de Matlab (`colormap jet`). Les sonagrammes, qui constituent des « empreintes acoustiques », sont très commodes pour caractériser des voix ou des instruments de musique.

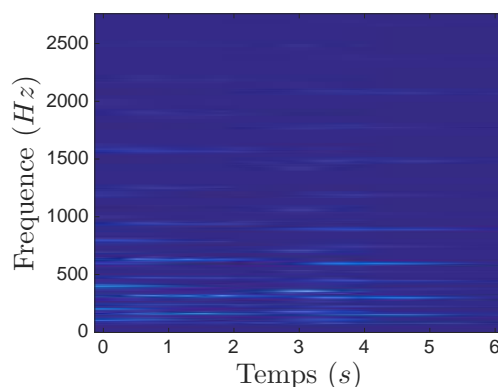


FIGURE 1 – Sonagramme d'un morceau de musique. À un instant donné et pour une fréquence donnée, le niveau de gris (affiché avec la carte des couleurs par défaut de Matlab) est proportionnel à la partie réelle du spectre.

Exercice 1 : transformation de Gabor d'un enregistrement musical

Lancez le script `musique.m`, qui lit un fichier audio au format WAV (vous pourrez bien sûr tester les autres fichiers WAV du répertoire Audio). Écrivez la fonction `T_Gabor`, appelée par le script `exercice_1.m`, qui permet de calculer et d'afficher la transformée de Gabor de cet enregistrement sonore. Ce script utilise : la fonction `fftshift`, qui a déjà été vue dans le TP10 ; la commande `axis xy`, qui oriente les ordonnées vers le haut.

Comme la transformation de Fourier est inversible et que les fenêtres utilisées pour le calcul de la transformation de Gabor forment une partition temporelle du signal, il est possible d'inverser cette dernière transformation à l'aide de la transformation de Fourier inverse. C'est ce que fait le script `exercice_1.m` en guise de vérification.

En revanche, l'inversion devient impossible si seule la partie réelle de la transformée de Gabor est conservée. Modifiez la fonction `T_Gabor` de manière à évaluer l'impact de cette perte d'information sur le son restitué (vous pourrez également évaluer cet impact lorsque seul le module complexe de la transformée de Gabor est conservé).

Exercice 2 : calcul du sonagramme complexe

Plutôt que de perdre soit la partie imaginaire, soit la phase de la transformée de Gabor, une autre façon de condenser l'information consiste à supprimer les coefficients du spectre correspondant aux fréquences négatives, ainsi que ceux qui correspondent aux plus hautes fréquences. Cette perte d'information est elle aussi irréversible, mais la différence entre le son original et le son restitué n'est pas forcément perceptible par l'oreille humaine. En effet, l'oreille « moyenne » est sensible aux fréquences comprises entre 20 *Hz* (sons graves) et 20000 *Hz* (sons aigus), mais elle ne peut discriminer deux fréquences voisines que jusqu'à 4000 *Hz* environ.

Écrivez la fonction `calcul_S`, appelée par le script `exercice_2.m`, qui doit retourner le *sonagramme complexe* du signal original, c'est-à-dire une sous-matrice `S` de `TG` constituée d'une partie des premières lignes de `TG`, ainsi que le taux de compression correspondant. Pour évaluer l'impact de cette perte d'information sur le son restitué, les données manquantes dans `TG` sont complétées par des zéros. Quelle est la valeur maximale du taux de compression pour laquelle vous jugez que le son restitué est fidèle à l'original ?

Exercice 3 : compression audio

La compression audio permet d'atteindre des taux de compression de l'ordre de 10, voire très supérieurs. Pour que le taux de compression du script `exercice_2.m` vaille 8, il faut que la proportion de fréquences positives conservées soit égale à 0,25, auquel cas vous constatez, même en n'étant pas musicien, que le signal restitué est extrêmement dégradé. Le principe de la compression audio est légèrement différent : il consiste à ne conserver, dans le sonagramme complexe, qu'une faible proportion des coefficients les plus élevés.

Le script `exercice_3.m` est censé détecter, à chaque mesure, les `n` coefficients du sonagramme complexe les plus élevés (au sens du module). Écrivez la fonction `calcul_S_max` appelée par ce script, qui retourne deux sous-matrices de `TG` comportant exactement `n` lignes : `indices_S_max` contient, à chaque mesure (donc sur chaque colonne), les indices des `n` coefficients les plus élevés (en module), et `S_max` les valeurs de ces coefficients. **Attention** : contrairement à l'exercice 2, le sonagramme complexe doit être calculé sans élimination d'aucune fréquence positive.

Testez le script `exercice_3.m` pour différentes valeurs du paramètre `n`. Vous constatez qu'en procédant ainsi, le taux de compression du signal audio peut atteindre des valeurs relativement élevées, sans que la dégradation du signal soit perceptible. Vous venez de réaliser un système de compression audio analogue à celui de la compression MP3 (en un peu moins sophistiqué).