TP10 – Compression audio

Transformation de Gabor d'un signal acoustique

La première *représentation temps-fréquence* d'un signal temporel a été proposée en 1946 par Gabor, physicien hongrois ayant obtenu le prix Nobel en 1971 pour l'invention de l'holographie, bien avant celle de Morlet et Grossmann en 1984, connue sous le nom de *transformation en ondelettes*. La *transformation de Gabor*, qu'on appelle également *transformation de Fourier locale* ou *transformation de Fourier à fenêtre glissante*, 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, c'est le plus souvent une « fonction créneau » qui est utilisée. La transformée de Gabor $\mathrm{TG}(x)$ d'un signal acoustique *continu* x s'écrit alors :

$$\mathrm{TG}\{x\}(t,f) = \mathrm{TF}\{x\,w_t\}(f)$$

où w_t désigne la fenêtre glissante, qui peut être positionnée à un instant t variable. Par conséquent, alors que le signal d'origine x ne dépend que de t, sa transformée de Gabor $\mathrm{TG}\{x\}$ est une fonction de deux variables réelles : le temps t et la fréquence f. La transformation de Gabor transforme donc la fonction réelle unidimensionnelle x en une fonction complexe bidimensionnelle $\mathrm{TG}\{x\}$, obtenue par concaténation de spectres instantanés.

La représentation d'une transformée de Gabor *discrète* se fait dans le plan : l'axe des abscisses représente le temps t, l'axe des ordonnées la fréquence f. Elle forme une image dont le niveau de gris est proportionnel au module complexe. Une colonne de cette image constitue le spectre instantané du signal $\mathbf{x}\,\mathbf{w}_t$ à un instant t. Le nombre de lignes de l'image est donc égal au nombre d'échantillons contenus dans $\mathbf{w}_t$. Quant au nombre de colonnes, il est égal au nombre de positions de la fenêtre glissante, qui dépend de la valeur du décalage. Il est courant de choisir un décalage égal à la largeur de la fenêtre, lorsque w_t est une fonction créneau, mais cela n'a rien d'obligatoire.

Une autre représentation temps-fréquence d'un signal sonore, appelée *sonagramme*, est obtenue en ne conservant que les coefficients de Fourier correspondant aux fréquences positives, car le module complexe d'une transformée de Gabor constitue une fonction « presque paire », relativement à la variable f. En outre, les coefficients de Fourier correspondant aux fréquences audibles les plus élevées sont également retirés, car il est admis que « la majeure partie de l'information acoustique réside dans les basses fréquences ». L'image résultante s'appelle un *sonagramme* (ou *spectrogramme sonore*) : la figure 1 montre un exemple de sonagramme, pour lequel seules les fréquences inférieures à 4000 Hz ont été conservées.

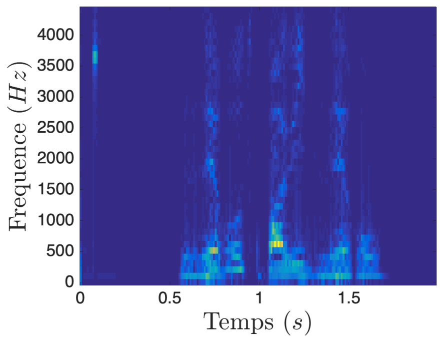

Figure 1 – Sonagramme d'un signal de parole. À un instant t donné, et pour une fréquence f donnée, le niveau de gris (affiché ici en fausses couleurs) est proportionnel au module complexe du spectre instantané.

Exercice 1 : calcul de la transformée de Gabor

Commencez par lancer le script `lecture`, qui lit un enregistrement sonore au format `WAV`. Vous pourrez bien sûr par la suite tester d'autres fichiers audio, par exemple ceux du répertoire `Audio`.

Écrivez la fonction `t_Gabor`, appelée par le script `exercice_1`, permettant de calculer et d'afficher la transformée de Gabor du signal passé en paramètre.

Remarques importantes :

- Les colonnes de la matrice `TG` 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. La commande `fftshift(TG)`, qui apparaît dans `exercice_1`, permet d'inverser verticalement les deux moitiés de la matrice `TG`. Après inversion, les fréquences correspondent à l'ordre « naturel » $f = -f_{\max}, \ldots, f_{\max}$.
- 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'orientation de l'axe des ordonnées 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 `t_Gabor` sans boucle `for`.
- La longueur d'un signal n'a aucune raison d'être un multiple du nombre d'échantillons contenus dans la fenêtre glissante. La dernière partie du signal, qui correspond à la dernière position de la fenêtre glissante, est donc généralement plus courte que les autres. Plutôt que de la compléter par des zéros, il est conseillé de la supprimer.

Dans le script `exercice_1`, les décalages successifs de la fonction créneau utilisés pour le calcul de la transformée de Gabor sont choisis de manière à former une partition du signal. La transformation de Fourier étant inversible, cette transformée de Gabor doit permettre de restituer le signal acoustique d'origine sans aucune distorsion. En revanche, cela n'est plus le cas si l'on ne dispose que de la partie réelle de la transformée de Gabor (perte de la partie imaginaire) ou que de son module complexe (perte de la phase). Le script `restitution_1` vous permet de comparer le son restitué dans ces trois cas de figure.

Exercice 2 : calcul du sonagramme

Plutôt que de perdre soit la partie imaginaire, soit la phase de la transformée de Gabor, une autre façon de compresser les données consiste à supprimer les coefficients de Fourier correspondant aux fréquences négatives. L'affichage effectué par le script `exercice_1` montre que le module de la transformée de Gabor est quasiment symétrique par rapport à l'axe des abscisses. Les coefficients de Fourier correspondant aux hautes fréquences peuvent également être supprimés. Cette double perte d'information est irréversible, mais la différence entre le son original et le son restitué est imperceptible. En effet, si l'oreille « moyenne » est sensible aux fréquences comprises entre 20 Hz (sons graves) et 20000 Hz (sons aigus), elle ne peut discriminer deux fréquences voisines que jusqu'à 4000 Hz environ.

Écrivez la fonction `sonagramme`, appelée par le script `exercice_2`, qui retourne le sonagramme du signal original, c'est-à-dire une sous-matrice `SG` de `TG` constituée des premières lignes de `TG`, à hauteur d'une certaine `proportion`, ainsi que le taux de compression atteint (n'oubliez pas qu'un complexe occupe deux fois plus de place qu'un réel). Il est bien sûr conseillé d'utiliser la fonction `t_Gabor` de l'exercice 1.

Le script `restitution_2` permet d'évaluer l'impact de cette perte d'information sur le signal sonore restitué (les données manquantes de `TG` étant complétées par des zéros). Quelle est la valeur maximale du taux de compression pour laquelle vous jugez que la restitution sonore demeure fidèle à l'original ?

Exercice 3 : compression acoustique

La compression acoustique permet d'atteindre des taux de compression de l'ordre de 10, voire très supérieurs. Or, pour que le taux de compression du script `exercice_2` vaille 8, il faut que la proportion de fréquences positives conservées soit égale à $1/8 = 0,125$, auquel cas le signal sonore restitué est très dégradé. Le principe de la compression acoustique est légèrement différent : il consiste à ne conserver, dans le sonagramme, qu'une faible proportion des plus grands coefficients de Fourier.

Le script `exercice_3` est censé détecter, sur chaque colonne du sonagramme, les `m` coefficients de Fourier les plus grands (en module). Écrivez la fonction `mp3`, appelée par ce script, qui retourne deux matrices `indices_max` et `valeurs_max` comportant `m` lignes chacune : pour chaque colonne, `indices_max` contient les numéros de lignes des `m` plus grands coefficients de Fourier (en module), et `valeurs_max` contient les valeurs de ces coefficients.

Le script `restitution_3` permet d'évaluer l'impact de cette perte d'information sur le signal sonore restitué. En testant différentes valeurs des paramètres `proportion` et `m`, vous constatez que le taux de compression du signal sonore peut atteindre une valeur avoisinant la dizaine, tout en garantissant une bonne restitution sonore. Vous venez de réaliser un système de compression acoustique analogue au MP3, dans une version très simplifiée à laquelle manque, en particulier, le codage des coefficients de Fourier sélectionnés.

Exercice 4 : stéganographie acoustique (exercice facultatif)

La *stéganographie*, du grec *steganos* (couvrir) et *graphō* (j'écris), désigne l'art de dissimuler un message dans un autre. La stéganographie ne doit pas être confondue avec la cryptographie. Dans ce dernier cas, on n'ignore pas la présence du message, mais on n'a pas la clé pour le décrypter. La stéganographie permet de faire passer un message secret, ou de marquer des données de manière invisible, afin de pouvoir repérer leur copie illégale.

La stéganographie acoustique consiste donc à cacher un enregistrement sonore dans un autre. Plutôt que de compresser les données en éliminant une certaine proportion des hautes fréquences, la place libérée peut être mise à profit pour stocker les coefficients de Fourier des basses fréquences d'un autre enregistrement sonore.

Le fichier `message_cache.wav`, que vous pouvez lire à l'aide du script `lecture`, contient un enregistrement sonore caché. À vous de le trouver en écrivant un script dédié, de nom `exercice_4`. Indication : pour rendre l'enregistrement sonore caché indétectable, ses coefficients de Fourier ont été divisés par 1000.