Travail pratique 1 : La transformée de Fourier locale

Objectifs

- Se familiariser avec l'environnement d'analyse et de développement rapide MATLAB;
- Comprendre l'espace fréquences-temps;
- Se familiariser avec le rapport entre les fréquences et les notes de musique;
- Rédiger un mini rapport d'expérimentation.

À remettre

1. Un rapport (fichier `.pdf`) dans lequel vous aurez mis :
   1.1. Les noms, prénoms et matricules pour chacun des deux coéquipiers;
   1.2. Les calculs et raisonnements théoriques qui vous mènent à votre code.
   1.3. Les fréquences et le nom des cinq premieres notes du fichier `Glock.wav`.
   1.4. Les résultats obtenus sous forme de capture d'écrans (signaux temporels, diagrammes graphiques...) ou tableaux;
2. Toutes les solutions programmées (`<nom>.m`) que vous vous êtes créées. C'est-à-dire, au minimum ces fichiers (vous pourriez en avoir plus) :
   2.1. `Solution.m` : code général contenant toutes les expérimentations que vous aurez faites (documentez soigneusement vos lignes de code)
   2.2. `STFT.m` : fonction TF locale
   2.3. `ISTFT.m` : fonction TF locale inverse
   2.4. `DispSTFT.m` : fonction qui permet de visualiser les coefficients
3. Le fichier audio (`.wav`) restauré par votre programmation.

Tâches à effectuer

1) Identifier des notes de musique

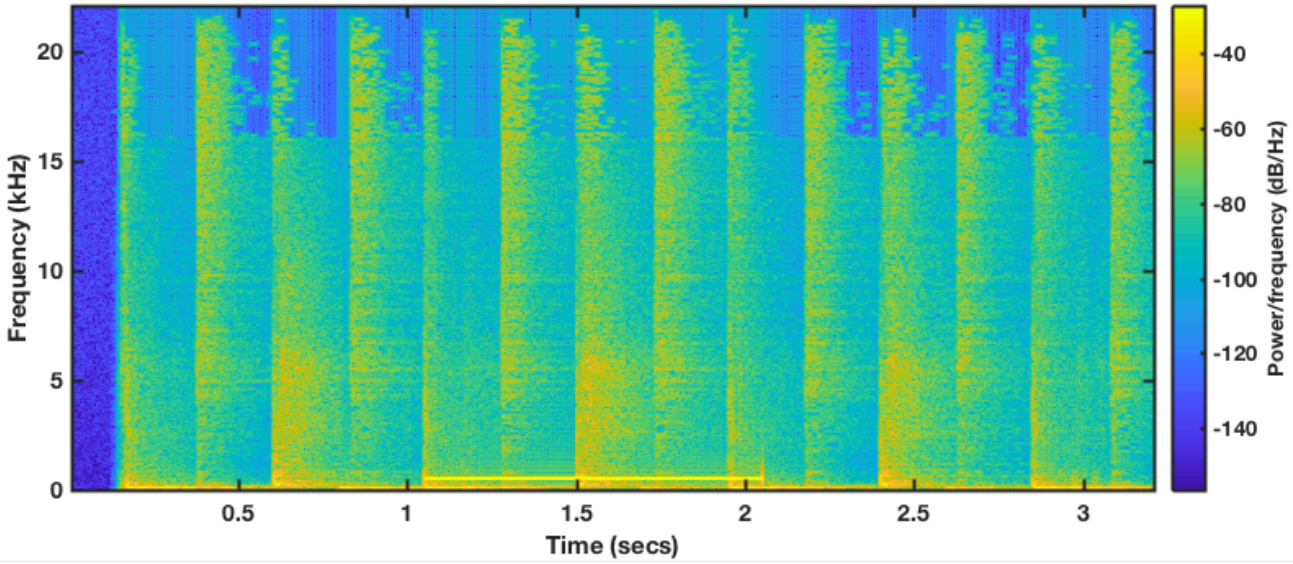
Le fichier `Glock.wav` contient une petite mélodie jouée sur un glockenspiel (allez voir ce qu'est un glockenspiel sur wikipédia). Vous devrez trouver les cinq premières notes de la mélodie.

Comment faire?

1. Calculer les coefficients de la TF locale et les afficher dans un format permettant d'identifier autant quelles sont les fréquences des notes que le moment où elles sont jouées.
2. Convertir chacune des fréquences identifiées en noms de note de la gamme tempérée.

2) Restaurer un fichier corrompu

Le fichier `Restore.wav` contient une sinusoïde non désirée. Voici un spectrogramme du son :



Notez la ligne jaune vers le bas-centre du spectrogramme. Cette ligne est le signal non désiré. Vous devrez enlever ce son parasite.en appliquant un masque binaire à la transformée de Fourier locale du signal.

Comment faire?

1. Calculer les coefficients de la TF locale et les afficher.
2. Construire un masque et multiplier les coefficients de la TF locale avec ce masque. Afficher les coefficients résultants.
3. Effectuer la TF locale inverse des coefficients résultants pour obtenir le signal corrigé.
4. Recalculer les coefficients du signal restauré et les afficher.

Compléments d'information

Comment lire un fichier audio

Pour lire un fichier, vous pouvez utiliser la fonction `audioread`. Le fichier `Glock.wav` est fourni en fichier supplémentaire.

```
>> [x, fs] = audioread('Glock.wav');
```

`x` contient les échantillons du signal audio et `fs` est la fréquence d'échantillonnage en Hz. Remarquez que `fs` est très important pour jouer le son correctement (entres autres). Utilisez `soundsc` pour écouter l'extrait.

```
>> soundsc(x, fs);
```

Faites des essais en modifiant `fs` pour voir comment cela affecte le son.

```
>> soundsc(x, fs/2);
>> soundsc(x, 2*fs);
```

Visualiser un signal audio

D'abord, utilisez la fonction de base `plot` :

```
>> figure;
```

```
>> plot(x);
>> axis tight;                    %enlève les espaces blancs inutiles
```

Notez que les unités sur l'axe des `x` sont des échantillons. Par contre, ces échantillons sont liés au temps. On aimerait mieux avoir les unités en secondes. Utilisez la fréquence d'échantillonnage pour afficher les unités en secondes :

```
>> t = (0:length(x)-1)/fs;
>> plot(t,x);
>> axis tight;
>> xlabel('seconds');
```

On peut voir que le signal dure environ 6 secondes. Notez qu'à partir de la représentation temporelle, on voit où les notes commencent et où elles terminent, mais il est difficile d'évaluer quelle note est jouée parce que cela prendrait le contenu en fréquences. Pour cela, il faut passer par la TF (vous pouvez tester la fonction `freqz`):

```
>> [H, W] = freqz(x);      %Allez voir l'aide de Matlab pour savoir à
   quoi correspondent H et W.
>> plot(W,abs(H));
>> axis tight;
```

L'unité des fréquences `W` est radians par échantillon, on aimerait mieux en Hz. Il faut utiliser la fréquence d'échantillonnage pour y arriver :

```
>> f = (fs/2)*W/pi;
>> plot(f/1000,abs(H));
>> axis tight;
>> xlabel('KHz');
```

L'unité des fréquences est bien en Hz. On voit des pics dans le graphique mais comment savoir quand une note commence? La fonction `STFT` (à coder) servira à cela.

```
>> win = hamming(50*fs/1000);        %fenêtre d'analyse de 50 msec.
>> hop = round(length(win)/4);       %le nb d'échantillons à sauter
>> X = STFT(x, win, hop);            %fonction à coder
```

`X` contient les coefficients de la TF locale. Comme c'est un domaine à 2 dimensions, on peut toujours la voir comme une image. On peut l'afficher avec la fonction `imagesc` (pour l'instant, n'affichons que l'amplitude).

```
>> imagesc( abs( X ) );
```

On peut plutôt afficher le log de l'amplitude, on verra mieux.

```
>> imagesc( log( abs( X ) ) );
```

C'est pas mal mieux, mais c'est encore difficile à interpréter parce que les axes ne sont pas correctement étiquetés (mauvaises unités) et on a n'a pas tenu compte de la symétrie fréquentielle (pour l'amplitude). Codez-vous une fonction DispSTFT qui affiche les coefficients de la TF locale où vous aurez les fréquences en Hz, le temps en secondes. Vous devriez pouvoir passer en paramètre une étendue de fréquences finie.

```
>> %préparation préliminaire
```

```
>> Fr = [0 8000];          %l'étendue des fréquences qu'on souhaite
   visualiser (en Hz)
>> clim = [-50 0];         %l'étendue des amplitudes qu'on souhaite
   visualiser (en decibels)
>> DispSTFT(X/max(abs(X(:))), fs, length(win), hop, Fr, clim);
   %fonction à coder
```

On peut maintenant voir quelle note (en fréquences) est jouée et quand. À partir des fréquences et des notes de la gamme, trouvez les 5 premières notes avec le temps où elles sont jouées du fichier `Glock.wav`. Vous trouverez un tableau des fréquences de la gamme tempérée sur wikipédia. N'oubliez pas que les fréquences de la même note sur un octave différent est relié par un rapport de 2.

Implémenter la TF locale

On veut utiliser une fenêtre pour trouver la TF sur une petite section du signal. Cette TF est une colonne de la TF locale. Puis, on glisse la fenêtre un peu plus loin (`hop`) et recalcule la TF pour remplir la colonne 2 de la matrice, ainsi de suite jusqu'à la fin du signal.

```
>> N = 40*fs/1000;                  %fenêtre de 40 ms
>> win = hamming(N);
>> hop = round(length(win)/4);      %le nb d'échantillons à sauter
>> F = fft(x(1:N) .* win );         %mult. pt à pt avec le signal
>> X = F;                           %F est la colonne 1 de X
```

La prochaine colonne est obtenue en glissant la fenêtre de `hop` échantillons :

```
>> F = fft( x( hop + (1:N) ) .* win ); %décalage de la fenêtre
>> X(:,2) = F;
```

Vous voyez la suite …

```
>> F = fft( x( 2*hop + (1:N) ) .* win );    %décalage de la fenêtre
>> X(:,3) = F;
```

Vous continuez jusqu'à ce que vous êtes à la fin de `x`.

Vous devez écrire une fonction appelée STFT qui sera appelée comme suit :

```
>> X = STFT(x, win, hop);           %vous passez la fenêtre et le hop
   déjà calculés
```

Inverser la TF locale

La représentation locale de la TF est utile car elle permet d'obtenir des effets intéressants via des opérations simples sur ses coefficients. Donc, quand on traite un signal avec la TF locale, l'idée est de calculer les coefficients, les modifier, puis faire la TF local inverse pour obtenir le signal modifié dans le domaine temporel. Voilà comment inverser la TF locale.

Rappel : Les colonnes de X contiennent les TF locales des segments de x (fenêtré). Comme on peut facilement inverser la TF, on peut reprendre les coefficients et revenir à la version fenêtrée du signal x :

```
>> y1 = ifft( X(:,1) );     %Les premiers N échantillons (fenêtrés)
>> y2 = ifft( X(:,2) );     %Les N échantillons après hop fenêtrés
```

```
>> y3 = ifft( X(:,3) );     %Les N échantillons après 2*hop fenêtrés
```

Tentons de reconstruire `x` en combinant les `y` comme suit :

```
>> y = zeros(N + 2*hop, 1);          %initialisation à 0
>> y(1:N) = y1;                      %Les N premiers échantillons
>> y( hop + (1:N) ) = y( hop + (1:N) ) + y2;      %décalage de hop
>> y( 2*hop + (1:N) ) = y( 2*hop + (1:N) ) + y3;  %décalage de hop
```

On voudrait avoir `y` égal aux premiers échantillons de `x` mais ce n'est pas le cas. Il faut tenir compte du fenêtrage. `y` peut-être vu comme une version masquée de `x`. Cette version masquée peut être construite, et sera similaire à `y`.

```
>> mask = zeros(N + 2*hop,1);
>> mask(1:N) = win;
>> mask( hop + (1:N) ) = mask( hop + (1:N) ) + win;
>> mask( 2*hop + (1:N) ) = mask( 2*hop + (1:N) ) + win;
>>
>> z = x( 1:(N+2*hop) ) .* mask;     %z devrait être égal à y
>>
>> max( abs (y - z) )          %vérification : devrait être presque 0
  ans =
     1.1102e-16
```

Écrivez une fonction appelée ISTFT qui prends en entrée les coefficient de la TF locale, la fenêtre utilisée pour calculer celle-ci et le `hop`. Vérifiez que votre fonction retrouve correctement le signal original si les coefficients ne sont pas modifiés.