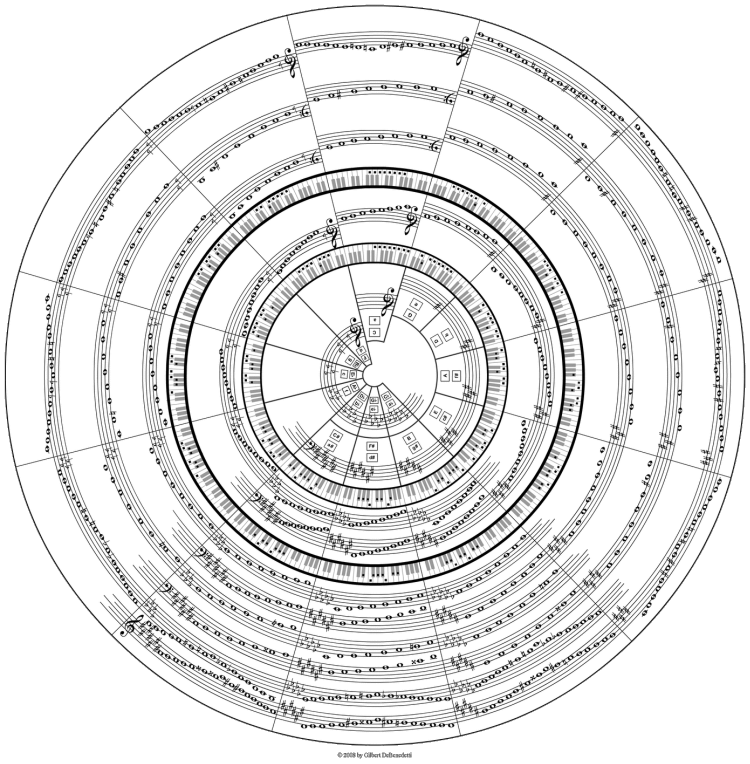
INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Département Génie Mathématique

Projet de transcription de musique

Projet semestriel – semestre 8



Rand ASSWAD
Ergi DIBRA
Yuge SUN

A l'attention de :
Mme. Natalie FORTIER

11 juin 2018

Contents

"La vie sans musique est tout simplement une erreur, une fatigue, un exil." — **Friedrich Nietzsche**

1 Introduction

La musique peut être considérée comme la première langue parlée. Bien que la musique soit un art, sa perception est un effort artistique aussi bien que scientifique.

En effet, l'humain cherchait à caractériser la musique afin de pouvoir la conserver et de la partager. Cette science a commencé par l'écriture des chants hourrites sur des tablettes d'argile extraite d'Ougarit (actuellement Latakié, Syrie) qui remontent approximativement à 1400 av. J.-C. [Wikipédia, 2018]

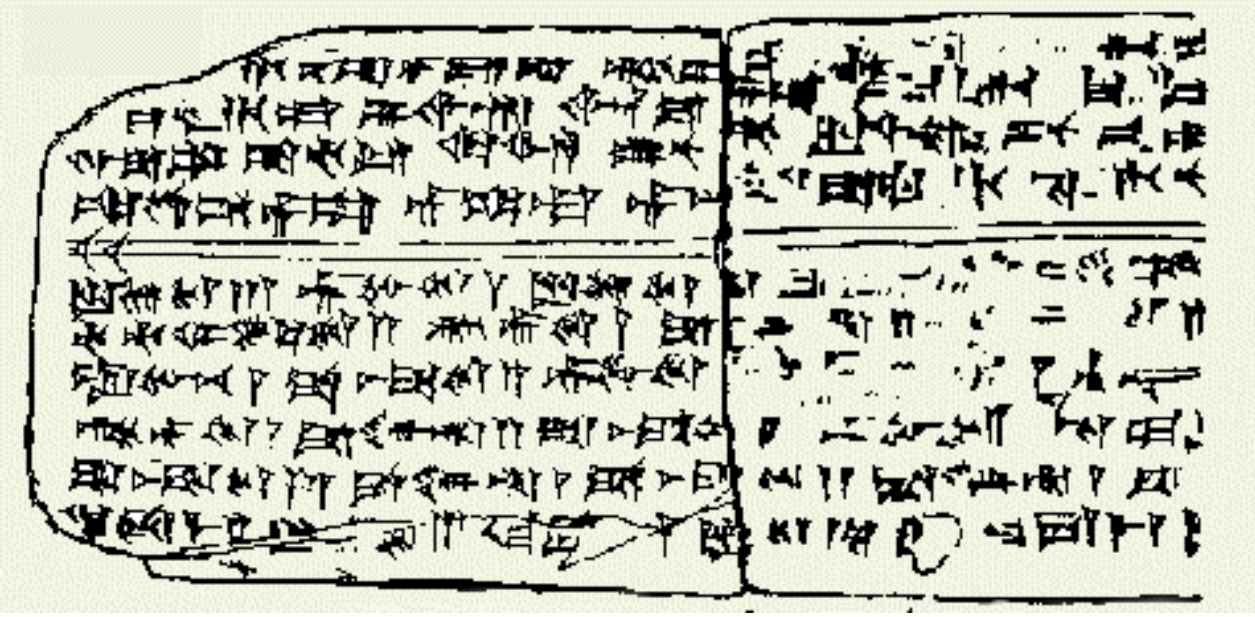


Figure 1: Dessin de la tablette de l'hymne à Nikkal (chant hourrite)

La notation musicale a évolué au cours de l'histoire, le système occidental développé au moyen âge appelé "*solfège*" est adopté aujourd'hui par tous les musiciens du monde avec quelques variations. La transcription d'une œuvre musicale est appelée une *partition*.

Ce système de notation est fidèle à la perception humaine du son, vis-à-vis la linéarité de fréquences et la reconnaissance du rythme. En effet, on tappe les pieds suivant le rythme de la musique sans aucune connaissance musicale, et la majorité de personnes est capable de chanter une mélodie sans lire sa partition.

En revanche, un signal sonore contient les mêmes informations sous une forme différente; l'espace de fréquences n'est pas linéaire et le rythme n'est pas reconnaissable facilement.

Dans ce projet, nous avons étudié et implémenté des méthodes d'interprétation de signaux sonores dans l'objectif de pouvoir produire une partition de musique à partir d'un son enregistré.

Des nombreuses recherches ont été effectuées sur ce sujet et les résultats obtenus sont certainement intéressants. Malheureusement, ces méthodes ne traitent que des cas particuliers.

Nous nous sommes intéressés par ce domaine d'applications car il porte un fort potentiel dans le futur. Ce projet nous a permis de faire notre premier pas dans ce domaine et de mettre en place nos compétences en mathématiques, en informatique et en musique.

2 Signaux sonores

2.1 Musique & Son

L'humain fait le lien entre la musique et le son dans son cerveau instantanément sans le moindre d'efforts. Néanmoins, nous voudrions établir ce lien de façon scientifique.

Un son possède des les caractéristiques suivants:

- Hauteur tonale (fréquence)
- Durée
- Intensité (énergie)
- Timbre (source sonore)

La musique se caractérise par:

- **La mélodie:** la suite de phrases ou motifs sonores monophones
- **L'harmonie:** l'ensemble de son différents simultanés
- **Le rythme:** la suite de durées du son
- **La nuance:** l'intensité relative du son
- **Le timbre:** la nature du son / son source / son empreinte

Nous allons expliquer dans ce projet les notions de base de ce lien.

2.2 Son harmonique

Le son d'un résonateur acoustique comme une chorde ou une colonne d'air est une onde stationnaire. On dit que tel son évoque un **pitch défini**. Dans le cas des instruments de percussion, le son présente une *inharmonicité*. On dit que tel son évoque un **pitch indéfini**. Dans ce projet on ne s'intéressera qu'au sons harmoniques de pitch défini.

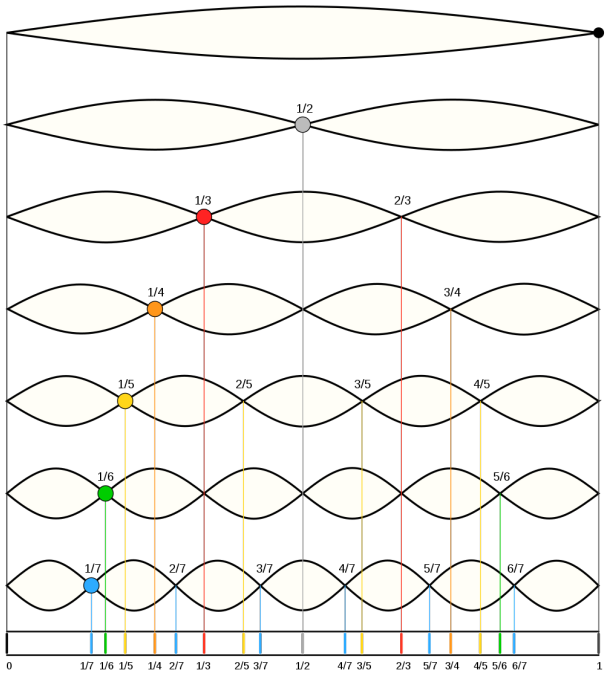


Figure 2: Les harmoniques d'une chorde vibrante

Un signal sonore de pitch défini, est une série harmonique de sons purs, représenté par des ondes sinusoïdales dont les fréquences sont des multiples **entiers** d'une fréquence dîte la **fondamentale** (où le **pitch**) notée f_0.

$$x(t) = \sum_{k \in \mathbb{N}} A_k \cdot \cos(2\pi k f_0 t)$$

où A_k est l'amplitude de la k$^{\text{ème}}$ harmonique.

On cherche donc à indentifier f_0 dans un signal harmonique donnée.

2.3 Discrétisation et échantillonnage

La numérisation d'un signal consiste à prélever des valeurs du signal à intervalles définis. Les valeurs obtenues sont appelées des *échantillons*.

La *période d'échantillonnage* T_s est l'intervalle de temps entre deux échantillons, on définit $f_s = \frac{1}{T_s}$ le nombre d'échantillons prélevés par secondes, f_s est dît *fréquence d'échantillonnage* ou en anglais **sample rate**.

On note $x[n] = x(t_n)$ où $t_n = n \cdot T_s = \frac{n}{f_s}$. Dans le reste du projet, on notera toujours $[\cdot]$ les valeurs discrètes.

L'échantillonnage d'un signal consiste à choisir une fréquence d'échantillonnage sans perdre de valeurs importantes du signal. En traitement de signaux sonores, f_s est souvent égale à $44.1kHz, 22.05kHz, 16kHz$, ou $8kHz$. La numérisation d'un signal dépend aussi d'autre facteurs comme le *bit depth* (i.e. le nombre de bits pour stocker chaque échantillon), mais nous ne nous intéressons pas par les détails; les bases de l'échantillonnage de signaux sont expliquées et démontrées par le théroème d'échantillonnage de **Nyquist-Shannon**.

2.4 La transformée de Fourier (FT)

La transformée de Fourier se définit par:

$$\hat{x}(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-2\pi j f t} \mathrm{d}t$$

Cette transformation permet d'identifier la fréquence d'une fonction périodique. En effet, La transformée de Fourier représente l'intensité d'une fréquence dans un signal, donc ses pics correspondent aux fréquences du signal.

Comme la transformée de Fourier est linéaire, la transformée d'un signal harmonique produit plusieurs pics.

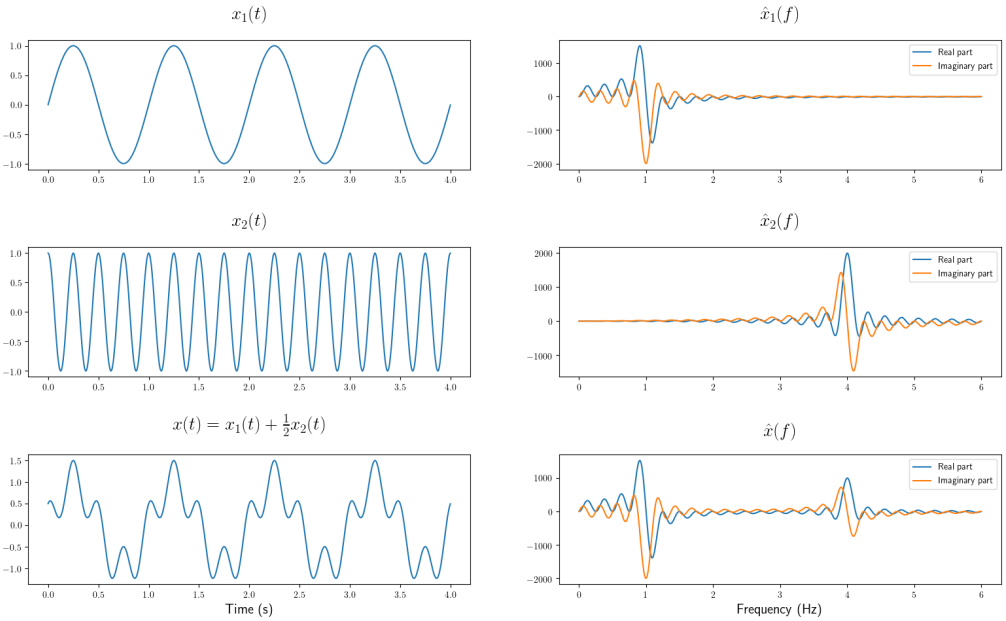


Figure 3: Linéarité de la transformée de Fourier

2.5 La transformée de Fourier discrète (DFT)

Soit N le nombre d'échantillons pris sur l'intervalle $[0, t_{\max}[$, soit f_s la fréquence d'échantillonage. Pour $n = 0, 1, \ldots, N-1$ on a:

$$\begin{aligned}\hat{x}(f) &= \int_0^{t_{\max}} x(t) \cdot e^{-2\pi j f t} \mathrm{d}t \\ &= \lim_{f_s \to \infty} \sum_{n=0}^{N-1} x(t_n) \cdot e^{-2\pi j f t_n} \\ &= \lim_{f_s \to \infty} \underbrace{\sum_{n=0}^{N-1} x[n] \cdot e^{-2\pi j f \frac{n}{f_s}}}_{\hat{x}[f]} \\ &= \lim_{f_s \to \infty} \hat{x}[f]\end{aligned}$$

La DFT de $x[n]$ se définit donc par:

$$\hat{x}[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-2\pi j k \frac{n}{f_s}}$$

Remarque: La DFT se calcule souvent matriciellement pour économiser les calculs. De plus, dans le cas où $N = 2^p, p \in \mathbb{N}$ on calcule la transformée de Fourier rapide (FFT) qui utilise le symétrie pour minimiser le nombre de calculs.

2.6 Fenêtrage

Nous avons souvent besoin de traiter le signal sur une durée limitée, on définit donc une fonction à support compact w et on étudie le produit de convolution du signal avec la fenêtre.

Voici quelques exemples de fenêtres:

- Fonction rectangulaire:

$$\mathrm{rect}_{[0,T]}(t) = \begin{cases} 1 & \text{si } t \in [0,T] \\ 0 & \text{sinon} \end{cases}$$

- Fenêtre Hann:

$$w(t) = \sin^2\left(\frac{\pi t}{T}\right) \cdot \mathrm{rect}_{[0,T]}(t)$$

- Fenêtre Welch (fenêtre parabolique):

$$w(t) = 1 - \left(\frac{2t - T}{T}\right) \cdot \mathrm{rect}_{[0,T]}(t)$$

Nous avons choisi d'utiliser la fenêtre de Hann dans notre projet car elle attenue le phénomène **aliasing** qui rend les signaux indisntinguables lors de l'échantillonnage.

$$w[n] = \sin^2\left(\frac{\pi n}{N-1}\right) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

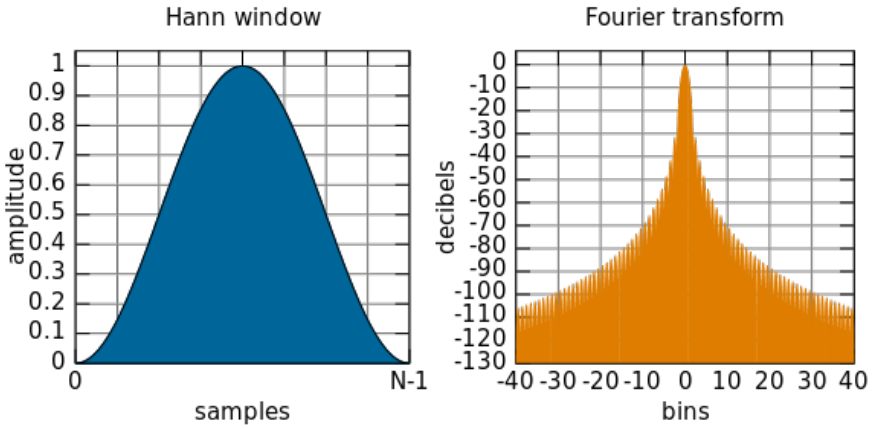


Figure 4: La fenêtre Hann est sa transformée de Fourier

2.7 La transformée de Fourier à court terme (STFT)

La transformée de Fourier nous permet d'obtenir les fréquences d'un signal harmonique. Or, la fréquence d'un signal peut changer en fonction du temps, on voudrait donc avoir la transformée de Fourier en fonction de la fréquence *et* du temps.

La transformée de Fourier à court terme $X(t, f)$ est la transformée de Fourier de x sur une fenêtre glissante w centrée en t (i.e. $w(\tau - t)$).

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) \cdot w(\tau - t) \cdot e^{-2\pi j f \tau} \mathrm{d}\tau$$

De même, la STFT discrète se définit:

$$X[n, k] = \sum_{n=0}^{N-1} x[m] \cdot w[m - n] \cdot e^{-2\pi j k \frac{m}{f_s}}$$

2.8 Spectrogramme

Le spectrogramme permet de visualiser les changement de fréquences en fonction du temps, il se définit par:

$$S(t, f) = |X(t, f)|$$

Remarque: Si on voudrait visualiser la puissance spectrale d'un signal, on prend le carré de la module de la STFT.

2.9 Méthodes d'analyse

Ils existent deux types d'analyse de signaux harmoniques:

1. **Analyse temporelle:** il s'agit d'étudier le signal sans passer par la transformée de Fourier.
2. **Analyse fréquentielle/spéctrale:** il s'agit d'étudier le spectre du signal (sa transformée de Fourier).

On obtient toujours des meilleurs résultats par l'analyse fréquentielle, mais la transformée de Fourier demande un calcul coûteux. Dans le cas d'un signal simple (e.g. percussion, rythmique, etc) on privilégie l'analyse temporelle. Sinon, dans le cas d'un signal complexe (e.g. instruments mélodiques, signal polyphône, etc) l'analyse fréquentielle est privilégié.

2.10 Implémentation

On lit un fichier audio grâce à la class `source` qu'on a défini

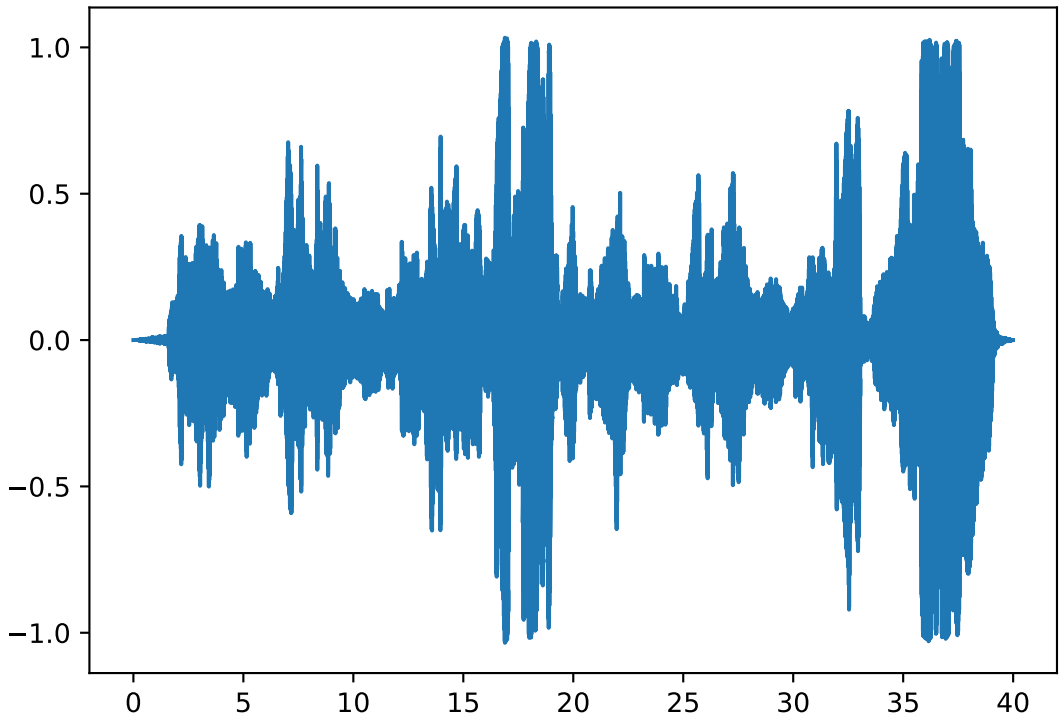
```
from muallef.utils import source
audio_file = project_path + "sounds/violin/violin-czardas-cut.wav"
sound = source(audio_file)
```

```
## /usr/lib/python3.6/site-packages/scipy/io/wavfile.py:273: WavFileWarning: Chunk (non-data) not unde
##   WavFileWarning)
```

```
x = sound.signal
fs = sound.sampleRate
t = sound.get_time()
```

On trace le signal à l'aide de la librairie `matplotlib`

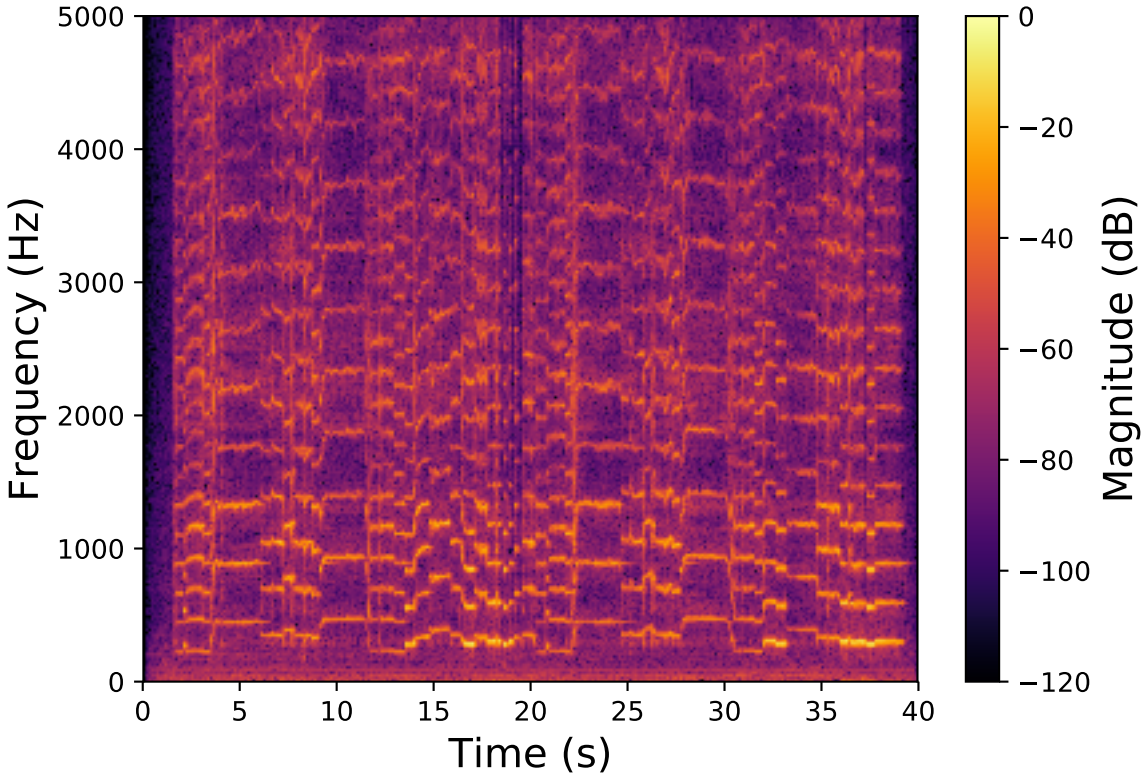
```
from matplotlib import pyplot as plt
plt.plot(t, x)
plt.show()
```




```
plt.clf()
```

On trace le spectrogramme du signal

```
from figures import plot_spectrogram
fig = plt.figure()
ax = fig.add_subplot(111)
ax, out = plot_spectrogram(signal=x, sample_rate=fs, axis=ax, color_map='inferno')
plt.show()
```




```
plt.clf()
```

3 Pitch

Dans cette partie, nous allons étudier des méthodes de reconnaissance de la fréquence fondamentale d'un signal.

3.1 YIN

L'algorithme de YIN *[Kawahara and de Cheveigné, 2002]* est une méthode robuste pour la reconnaissance du pitch dans le domaine temporel. Son principe est la séléction de fréquences candidates parmi toutes les fréquences détéctées sur l'intervalle de fenêtrage.

La méthode propose que l'expression $x(t) - x(t+\tau)$ atteint son minimum quand τ est égale à la période du signal (i.e. $\frac{1}{f_0}$). En diffinissant la fonction de différence à l'instant t fixé:

$$d_t[\tau] = \sum_{i=t+1}^{t+W} (x[t] - x[t+\tau])^2$$

où W est la taille de la fenêtre w, on appelle τ le *retard* (anglais: *lag*).

Par la suite, on calcule la fonction de la moyenne cumulative définie par:

$$d'_t[\tau] = \begin{cases} 1 & \text{si} \tau = 0 \\ d_t[\tau] / \frac{1}{\tau} \sum_{i=0}^{\tau} d_t[i] & \text{sinon} \end{cases}$$

Les candidats sont les minimums locaux de d'_t. On séléctionne les candidats avec $d'_t[\tau]$ inférieur à un seuil fixé (les auteurs recommandent un seuil de 0.1).

3.2 YIN spectrale

L'algorithme de YIN spectrale *[Brossier, 2007]* est une méthode qui utilise la même logique de l'algorithme de YIN et l'applique sur le spectre du signal.

La fonction de différence est définie par le carré de la différence spéctrale:

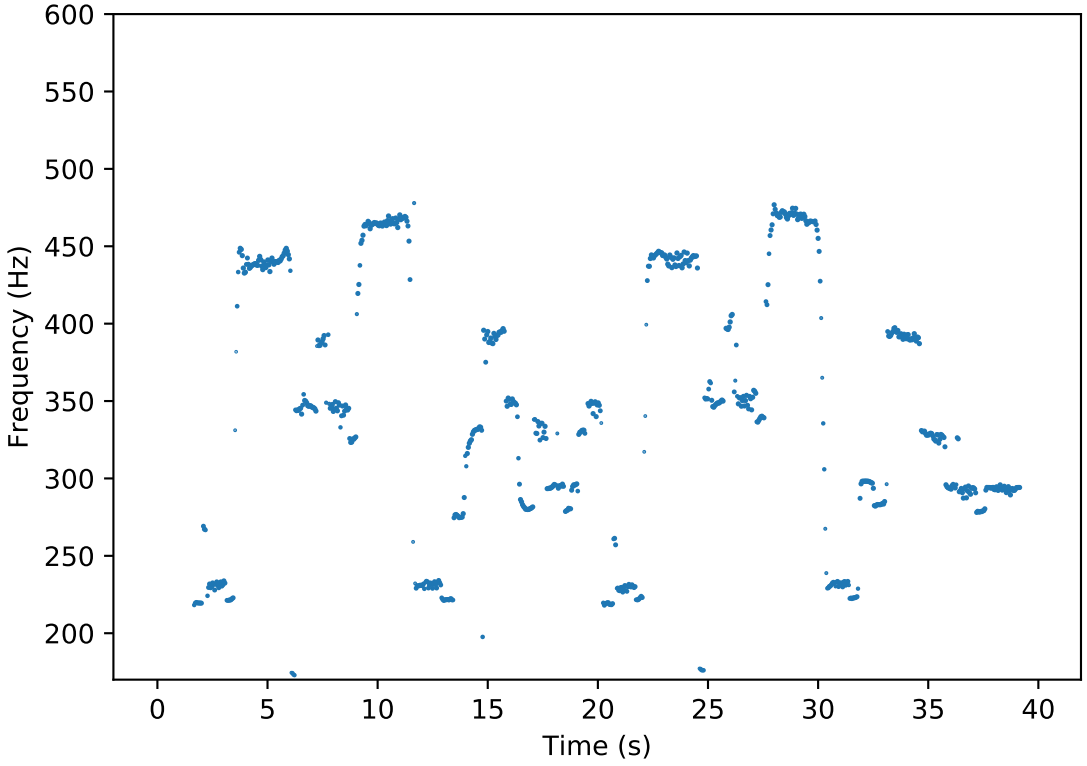
$$\begin{aligned} \hat{d}_t[\tau] &= \frac{4}{N} \sum_{k=0}^{\frac{N}{2}+1} |X[t,k]|^2 - \frac{2}{N} \sum_{k=0}^{\frac{N}{2}+1} |X[t,k]|^2 \cdot \cos\left(\frac{2\pi k\tau}{N}\right) \\ &= \frac{2}{N} \sum_{k=0}^{\frac{N}{2}+1} |X[t,k]|^2 \cdot \left(2 - \cos\left(\frac{2\pi k\tau}{N}\right)\right) \\ &= \frac{2}{N} \sum_{k=0}^{\frac{N}{2}+1} \left|\left(1 - e^{2\pi jk\tau/N}\right) X[t,k]\right|^2 \end{aligned}$$

La fonction de la moyenne cumulative se calcule de façon analogue à l'algorithme de YIN. L'algorithme cherche le minimum globale de cette dernière.

Cette méthode est la plus utilisée pour la détection de pitch car elle donne de très bonnes résultats et a pour complixité $O(n \log n)$

3.3 Implémentation

```
import numpy as np
from muallef import pitch
t, f, conf = pitch.detect(signal=x, sampleRate=fs, bufferSize=2048, method='yinfft')
# t: le temps en secondes
# f: la fréquence en Hz
# conf: le taux de confiance
size = np.power(conf, 3)
fig = plt.figure()
ax = fig.add_subplot(111, ylim=(170, 600))
ax.scatter(t, f, marker='o', s=size)
ax.set_xlabel("Time (s)")
ax.set_ylabel("Frequency (Hz)")
plt.show()
```

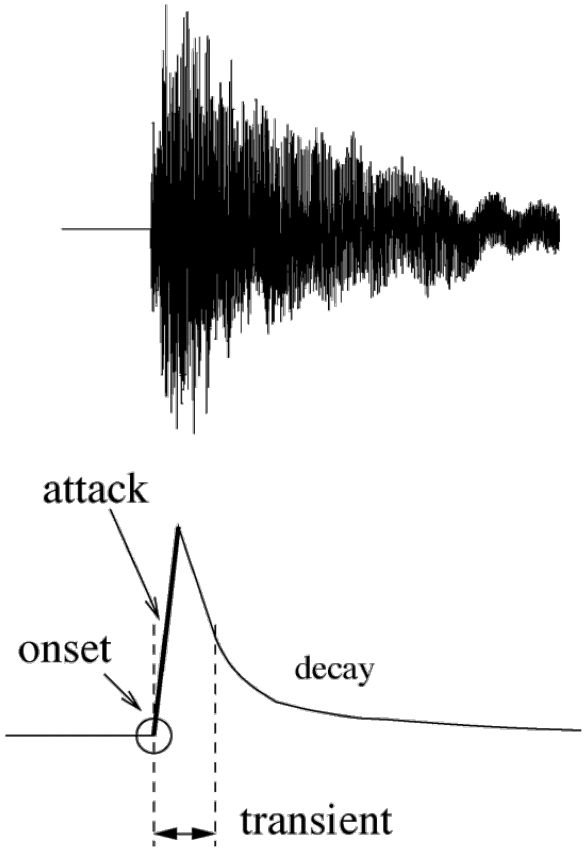

600
550
500
450
400
350
300
250
200
Frequency (Hz)
0
5
10
15
20
25
30
35
40
Time (s)

```
plt.clf()
```

4 Segmentation temporelle

L'étape fondamentale dans la reconnaissance du son est la segmentation temporelle. Il s'agit de trouver les frontières des objets sonores, c'est-à-dire:

- Le début de la note – dît ***onset***.
- La fin de la note – dît ***offset***.



Attack, transient, decay, onset IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 13, NO. 5, SEPTEMBER 2005

Cette étape dépend fortement sur le type du son produit; les instrument à chordes pincées (guitare, piano, oud, etc.) ont un profile différent de celui des instruments à cordes frottées (la famille du violon) ou de celle des instruments à vent.

Dans cette partie on expliquera les méthodes implémentés pour la reconnaissance du **onset**.

4.1 Méthode

La lecture scientifique nous a donné une méthode rigoureuse qu'on a simplifié pour obtenir des résultats rapidement.

Il s'agit de définir une fonction qui permet de quantifier la perturbation du signal à un moment donné, cette fonction est souvent appelée **Onset Detection Function** ou **Onset Strength Signal**, dans ce projet on fera référence à cette dernière par **Onset Detection Function** ou ***ODF***.

Théoriquement, les maximums locaux de l'ODF sont les onsets du signal, mais en pratique il s'agit d'un sous-ensemble de ces points. En effet, l'ODF est souvent très sensible et détectera la moindre des perturbations.

Ce problème pourra être résolu en définissant un seuil au dessous duquel aucun onset est considéré. Ils existent plusieurs méthodes pour définir tel seuil.

Soit un seuil fixe, ce qui minimise le coût des calculs au prix de la qualité des résultats. Soit de calculer un seuil variable, il s'agit de lisser la fonction ODF par des méthodes classiques comme la moyenne mobile.

La méthode consiste donc en trois étapes:

1. Calcul de l'**Onset Detection Function**.

2. **Thresholding**: calcul du seuil.
3. **Peak-picking**: la selection des onsets.

Une méthode heuristique est proposée par [Rosão et al., 2012] pour séléctionner les onsets, il s'agit de trouver les points t_n tels que, pour $a, b, \tau \in \mathbb{N}, \delta \in R_+$ fixés: - $x[n] = \max_{n+a \leq i \leq n+b} x[i]$ - $x[n] \geq \delta + \frac{1}{a+b+1} \sum_{n+a \leq i \leq n+b} x[i]$
- Si O est l'ensembles d'onsets, $\forall n, m \in O, |n - m| > \tau$

Cette méthode permet de vérifier que le point choisi est un maximum local et suffisament loin du point précédant, l'avantage de cette méthode est sa rapidité.

4.2 Onset Detection Function (ODF)

Ils existent plusieurs fonction de détéction d'onsets, on expliquera quelques unes qui se basent sur la STFT.

4.2.1 High Frequency Content (HFC)

Il s'agit de priviligier les fréquences élevées dans un signal.

$$D_{HFC}[n] = \sum_{k=1}^{N} k \cdot |X[n,k]|^2$$

[Masri, 1996]

4.2.2 Phase Deviation (Phi)

Il s'agit de calculer les différences de phases en dérivant l'argument complex de la STFT, on note

$$\varphi(t,f) = \arg(X(t,f))$$

$$\hat{\varphi}(t,f) = \text{princarg}\left(\frac{\partial^2 \varphi}{\partial t^2}(t,f)\right)$$

où

$$\text{princarg}(\theta) = \pi + ((\theta + \pi) mod(-2\pi))$$

donc la ODS de phase se calcule par la formule:

$$D_\Phi[n] = \sum_{k=0}^{N} |\hat{\varphi}[n,k]|$$

[Bello and Sandler, 2003]

Dans notre implémentation, nous avons approximé la dérivée partielle seconde de la phase par un schéma de Taylor d'ordre 2.

4.2.3 Complex Distance

Cette méthode permet de qualifier les changements spectraux du signal ainsi que les changements en phase. Il s'agit de calculer une prédiction du spectre du signal, et puis le comparer par sa valeur. On reprend la fonction calculée en $\hat{varphi}(t,f)$ de la méthode précédante. On définit la prédiction :

$$\hat{X}[n,k] = |X[n,k]| \cdot e^{j\hat{\varphi}[n,k]}$$

Donc la distance complexe se calcule:

$$D_{\mathbb{C}}[n] = \sum_{k=0}^{N} \left| \hat{X}[n,k] - X[n,k] \right|^2$$

[Duxbury et al., 2003]

4.3 Thresholding

Nous avons décidé de lisser la fonction ODF par une moyenne mobile echelonnées par la fenêtre Hann, il s'agit du produit de convolution de l'ODF avec la fonction Hann.

4.4 Résultats

Au premier lieu, nous avons effectués des tests sur des morceaux de violon joués par Rand et nous avons obtenus de très mauvaises résultats de segmentation temporelle. Nous avons ensuite essayé des morceaux plus facile joués par des musiciens plus expérimentés mais nous n'avons pas obtenu des meilleurs résultats.

On a donc considéré d'autres instruments, nottamment ceux à chordes pincées comme le piano et la guitare. Nous avons directement obtenu des très bons résultats.

Ceci s'explique par le fait que la famille de violon (chordes frottées) produit un son ***Legato***, la segmentation temporelle dépend donc moins sur l'énergie du signal (ou son spectre) et plus sur la fréquence fondamentale.

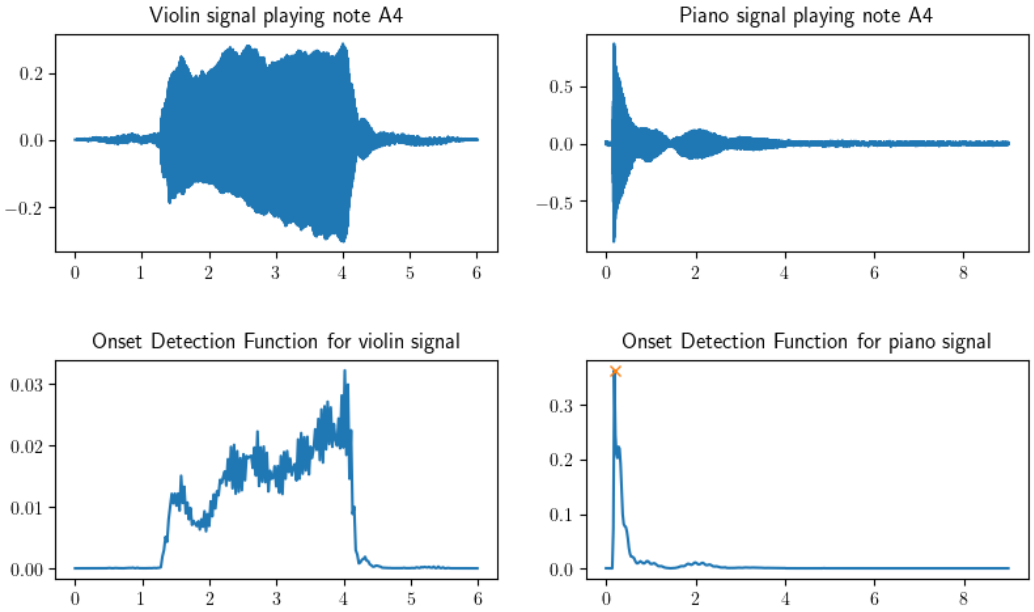


Figure 5: Comparaison entre les profiles temporels du violon et du piano

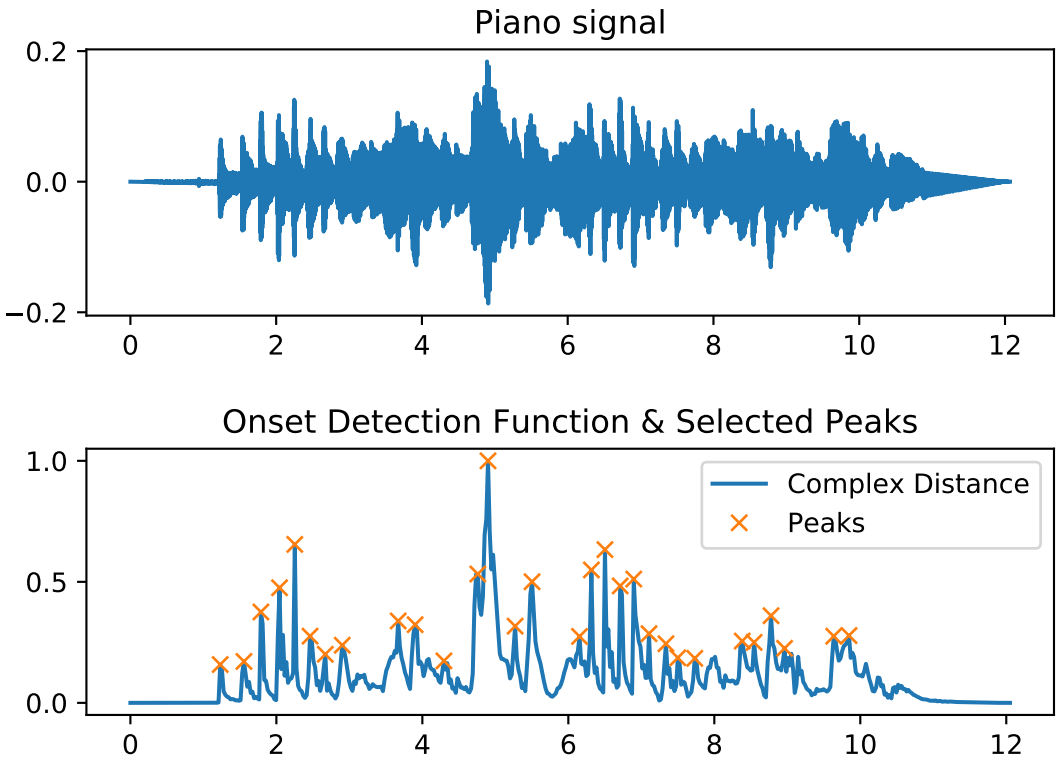
On voit bien que les onsets sont bien détectés dans le cas du piano. Voici le morceau fameux de Beethoven **Für Elise**.

```
from muallef.onset import onset_function
from muallef.onset.peak_picker import peak_pick
piano_sound = project_path + "sounds/piano/FurElise-mono.wav"
fur_elise = source(piano_sound)
```

```
## /usr/lib/python3.6/site-packages/scipy/io/wavfile.py:273: WavFileWarning: Chunk (non-data) not unde
##   WavFileWarning)
```

```
plt.clf()
fig = plt.figure()
ax1 = fig.add_subplot(211)
ax1.plot(fur_elise.get_time(), fur_elise.signal)
ax1.set_title("Piano signal")
ax2 = fig.add_subplot(212)
t, odf = onset_function(signal=fur_elise.signal, sampleRate=fur_elise.sampleRate,
```

```
                                windowSize=2048, method="complex", normalize=True)
ax2.plot(t, odf, label="Complex Distance")
onsets = peak_pick(odf, delta=0.05, wait=4)
ax2.plot(t[onsets], odf[onsets], marker="x", ls="", label="Peaks")
ax2.legend(loc="upper right")
ax2.set_title("Onset Detection Function & Selected Peaks")
plt.subplots_adjust(hspace=0.5)
plt.show()
```



5 Théorie de musique

La théorie de la musique permet, à la fois de réglementer la musique et de la libéraliser.

5.1 Notions de base

La **note** est le plus petit objet musical, porte un nom, et caractérise la hauteur tonal du son (fréquence). Dans le contexte d'un morceau musical, une note caractérise aussi la durée de cet objet.

Un **intervalle** se définit musicalement par l'écart de hauteur tonal entre deux notes. Scientifiquement, un intervalle est le ratio de fréquences fondamentales de deux notes.

On appelle ***octave*** l'intervalle correspondant au ratio $2:1$ Les notes d'un octave porte le même nom.

Une **échelle** musicale est une suite d'intervalles conjoints. Une **gamme** musicale est une suite de notes conjointes, la dernière répétant la première à l'octave.

L'**armure** — ou l'**armature** (*en:* **key signature**) — est un ensemble d'altérations réunies à la clé. Elle est composée soit exclusivement de dièses, soit exclusivement de bémols — en dehors du cas particulier constitué par le changement d'armure. Ces altérations correspondent à la tonalité principale des mesures suivant la clé.

Ils existent plusieurs théories de musique qui diffèrent principalement par la composition d'échelles et de gammes. Dans ce projet nous avons considéré la théorie de la musique occidentale basée sur l'accord tempéré.

5.2 Tonalité

5.2.1 Généralités

- L'unité de tonalité est le ***ton***.
- En musique classique, le plus petit intervalle est d'un ***demi-ton***.
- Un octave est composée de 6 tons, soit 12 demi-tons.
- L'échelle majeure classique est composée des intervalles: 1–1–½–1–1–1–½.
- La gamme majeure classique est composée de 7 notes distinctes (la $8^{\text{ème}}$ est à l'octave de la première).
- **Le dièse (#)** est une altération qui lève une note d'un demi-ton.
- **Le bémol (♭)** est une altération qui baisse une note d'un demi-ton.

5.2.2 Les notes

Les notes principales sont les touches blanches d'un piano. Les touches noirs d'un piano sont des notes altérées.

- Noms français: do-ré-mi-fa-sol-la-si
- Noms alphabétiques: C-D-E-F-G-A-B

Remarque: Certaines notes altérées sont des touches blanches (e.g. Mi#=Fa≡touche blanche), sans détailler sur les altérations composées (doubles dièses, doubles bémols).

5.2.3 Nomenclature

Ils existent plusieurs systèmes de nomenclature de notes de musique. Le système utilisé en France adopte les noms en termes de *Do-Ré-Mi-Fa-Sol-La-Si*. De plus, il existe un système basé sur l'alphabet latin : *C-D-E-F-G-A-B*. Les deux systèmes sont très utilisés, dans ce projet on utilisera le dernière pour simplifier.

Vu que les noms des notes se répètent au bout d'un octave, il faut distinguer une note LA de fréquence $440Hz$ d'une autre de fréquence $220Hz$ ou $880Hz$.

Le système de notation scientifique **Scientific Pitch Notation** identifie une note par sont nom alphabetique avec un nombre identifiant l'octave dans laquelle elle se situe, où l'octave commence par une note C. Par exemple la fréquence $440Hz$ représente A_4 sans ambiguité, et les fréquences $220Hz$ et $880Hz$ représentent les notes A_3, A_5 respectivement.

Dans le protocole **MIDI**, le notes sont représentées par un nombre entier, il permet de coder plus de 10 octave en partant de la note C_{-1}.

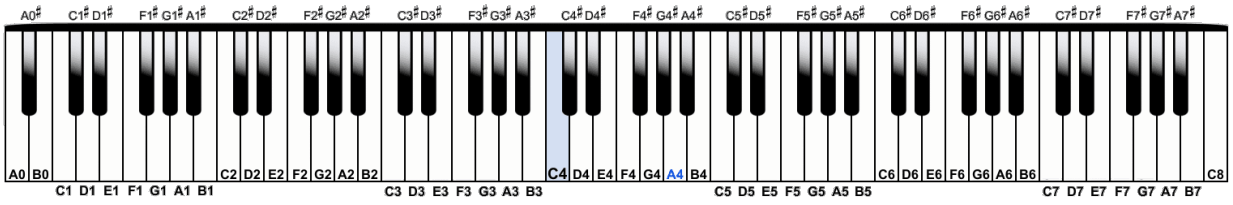


Figure 6: La notation scientifique sur un piano

5.2.4 Les gammes classiques

À chaque tonalité majeure est associée une tonalité en mode mineur, présentant la même armure de clef et appelée relative mineure.

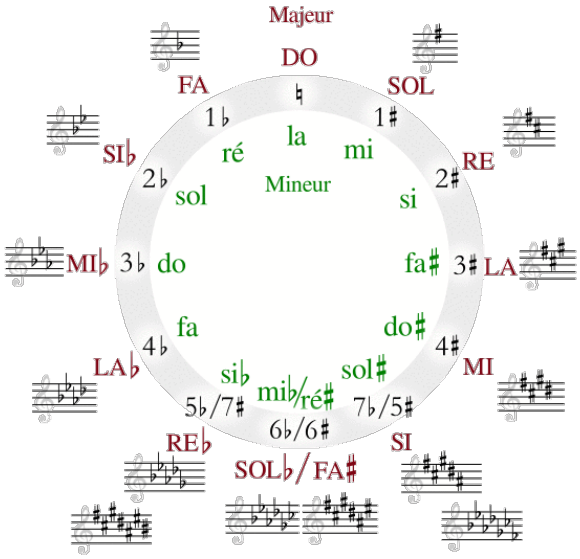


Figure 7: Le cycle des quintes

Pour simplifier, on va considérer la théorie de la musique occidentale basée sur l'accord tempéré (depuis le XVIII^{e} siècle). Dans ce cas, l'intervalle séparant la première et la dernière note d'une gamme est dite *octave*, une octave se divise en 12 écarts égales appelés *demi-tons*. La dernière note porte le même nom de la première dans la gamme.

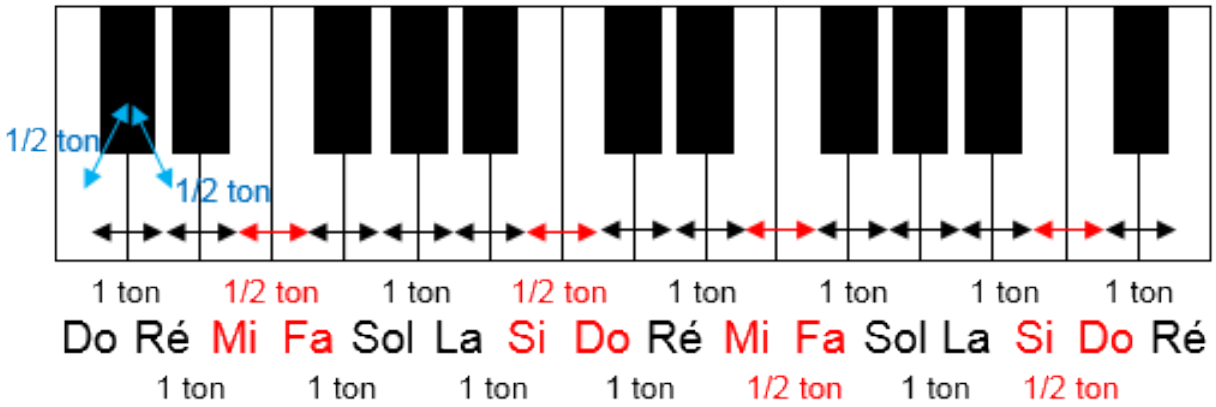


Figure 8: Les intervalles sur un piano

5.3 Tonalités & Fréquences

L'espace de notes est un espace linéaire discrèt, mais l'espace de fréquences est continue non-linéaire. Par exemple, la l'intervalle entre A3 et A4 est un octave, celui entre A4 et A5 également. En revanche, $f(A3) = 220Hz, f(A4) = 440Hz, f(A5) = 880Hz$, on voit bien que la relation est logarithmique. Le problème consiste à trouver une fonction qui associe les fréquences fondamentales obtenues avec des valeurs entières.

5.3.1 Le protocole MIDI

On voudrais savoir le rapport r de fréquences associé à un demi-ton, sachant que l'octave double la fréquence on peut conclure facilement:

$$r^{12} = 2 \Rightarrow r = 2^{1/12}$$

On souhaite ramener l'espace de fréquences $(\mathbb{R}, \times)$ à l'espace $(\mathbb{N}, +)$ tel que $\boxed{\text{demi-ton} \equiv 1}$. On définit donc une bijection

$$\forall f \in]0, \infty[, f \mapsto 12 \log_2 f$$

En arrondissant le résultat à la valeur entière la plus proche, on obtient un espace linéaire discrèt correspondant aux notes.

Il sera convenient d'obtenir les mêmes notes du protocole **MIDI** vu qu'il est très bien établi et très utilisé. Pour cela, on effectue une petite translation, en partant de la note de référence $A4 \equiv 69_{\text{MIDI}} \equiv 440Hz$.

$$\begin{cases} \varphi : f \mapsto 12 \log_2 f + c_{\text{ref}} \\ \varphi(440) = 69 \end{cases} \Rightarrow c_{\text{ref}} = 69 - 12 \log_2 440$$

Par conséquent, la bijection φ est définit par :

$$\varphi :]0, \infty[\to \mathbb{R} : f \mapsto 12 \log_2 f + c_{\text{ref}} \quad \text{avec } c_{\text{ref}} = 69 - 12 \log_2 440$$

On note $\bar{\varphi}$ la fonction définit par $\bar{\varphi}(f) = \lfloor \varphi(f) \rceil \in \mathbb{Z}$ où $\lfloor \cdot \rceil$ est la fonction d'arrondissement à l'entier le plus proche.

On peut donc obtenir les nombres MIDI de notes à partir des fréquences fondamentales grâce à la fonction $\bar{\varphi}$.

Néanmoins, le nombre MIDI n'est pas suffisant pour identifier une note, car certaines notes ont la même fréquence en accord tempéré et donc le même nombre midi (i.e. la même touche sur un piano), par exemple $\text{MIDI}(C\#) = \text{MIDI}(D\flat)$. Pour distinguer ces notes il est nécessaire de trouver la gamme du morceau.

5.4 Reconnaissance de la gamme/l'armature

Dans cette étude, on ne s'intéressera aux notes dans une octave. On introduit donc la fonction ψ :

$$\psi :]0, \infty[\to [0, 12[: f \mapsto \psi(f) \mod 12$$

De même, on définit la fonction $\bar{\psi}$ telle que $\bar{\psi}(f) = \lfloor \psi(f) \rceil$. On voit que $\text{Im}(\bar{\psi}) = \mathbb{Z}/12\mathbb{Z}$.

Note	C		D		E	F		G		A		B
$\bar{\psi}(f)$	0	1	2	3	4	5	6	7	8	9	10	11

En musique classique, ils existent 4 types de gammes, on ne s'intéressera qu'à un : *la gamme majeure*. Comme on l'a déjà dit, une gamme est caractérisée par sa première note et la suite des intervalles. Dans la gamme majeure, les intérvalles en fonction du ton sont : 1–1–½–1–1–1–½.

La gamme *Do/C Majeur* contient donc les notes {0, 2, 4, 5, 7, 9, 11}.

De même, la gamme *Sol/G Majeur* contient les notes {7, 9, 11, 0, 2, 4, 6}. Ces gammes diffèrent par une note, la note 5≡F est remplacée par la note 6 qui correspond à F# ou G♭. Dans le contexte du Sol Majeur, on sait que 6≡F# car la gamme contient déjà 7≡G.

On voit bien que l'identification de la gamme est *nécessaire* pour la distinction entre certaines notes.

Une gamme peut être alors identifié par son ensemble de notes qu'on notera G tel que $G \subset \mathbb{Z}/12\mathbb{Z}, |G| = 7$. On définit le vecteur $g \in \{0,1\}^{12}$ associé à G tel que

$$g_i = 1_G(i) = \begin{cases} 1 & \text{si } i \in G \\ 0 & \text{sinon} \end{cases}$$

On définit donc E l'ensemble de gammes majeures.

Soit F l'ensemble de fréquences fondamentales obtenues, soit $S = \bar{\psi}(F) \subset \mathbb{Z}/12\mathbb{Z}$.

On définit $p : Z/12\mathbb{Z} \to \mathbb{N} : n \mapsto$ le nombre d'occurances de n dans le morceau.

On note $p_{\max} = \max_{n \in S} p(n)$. On définit le vecteur $x \in [0,1]^{12}$ tel que $x_i = \frac{p(i)}{p_{\max}}$.

La gamme du morceau est alors la solution du problème d'optimisation

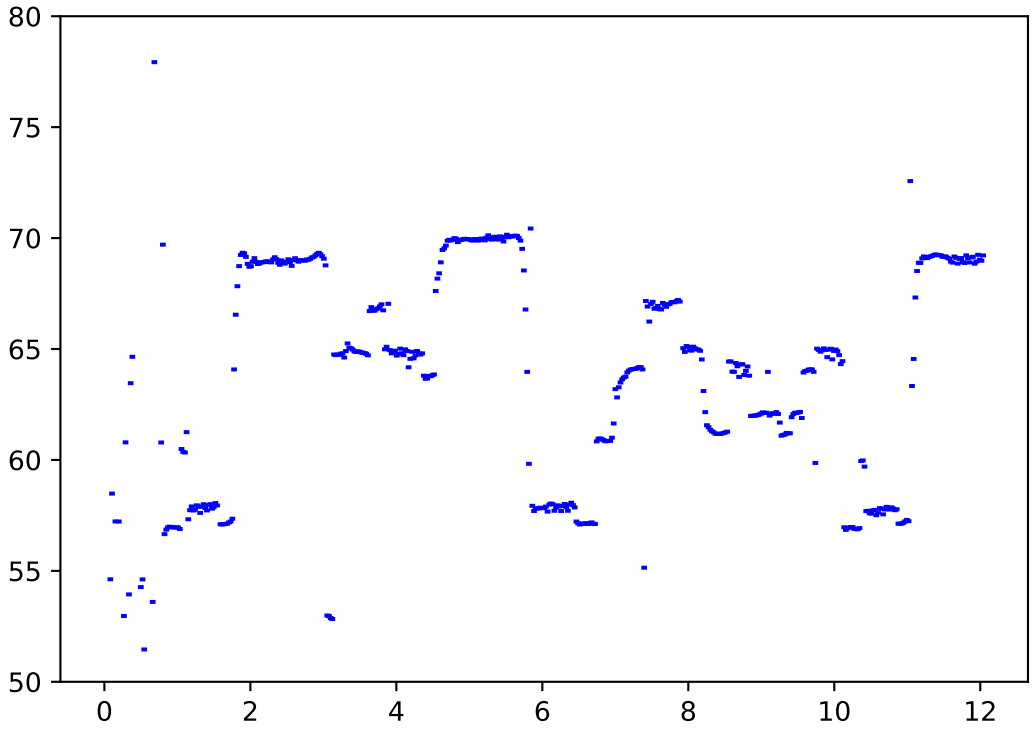
$$\min_{g \in E} \quad \|g - x\|$$

En musique classique, $|E| = 12$ donc le problème d'optimisation ne nécessite pas une résolution mathémtique avancée.

5.5 Implémentation

Nous pouvons tester cet algorithme sur notre morceau:

```
from muallef.utils.units import convertFreq
from muallef.notes import scale, name
from muallef.utils.io import plot_step_function
midi = convertFreq(f, "Hz", "MIDI")
plt.clf()
fig = plt.figure()
ax = fig.add_subplot(111, ylim=(50,80))
plot_step_function(t, midi)
plt.show()
```




```
base_note = scale.detect_scale(midi)
nb_alt, tone = scale.scale_signature(base_note)
print("Major scale: ", name.octave_note_name(base_note, tone))
```

```
## Major scale:  Fa
```

```
key_signature = ""
if tone > 0:
    for i in range(nb_alt):
        key_signature += name.octave_note_name(scale.sharp[i], tone) + " "
    key_signature += "\t/\t"
else:
    for i in range(nb_alt):
        key_signature += name.octave_note_name(scale.flat[i], tone) + " "
print("Key signature: ", key_signature)
```

```
## Key signature:  Si♭
```

En comparant avec la partition original de la fameuse Csárdás:

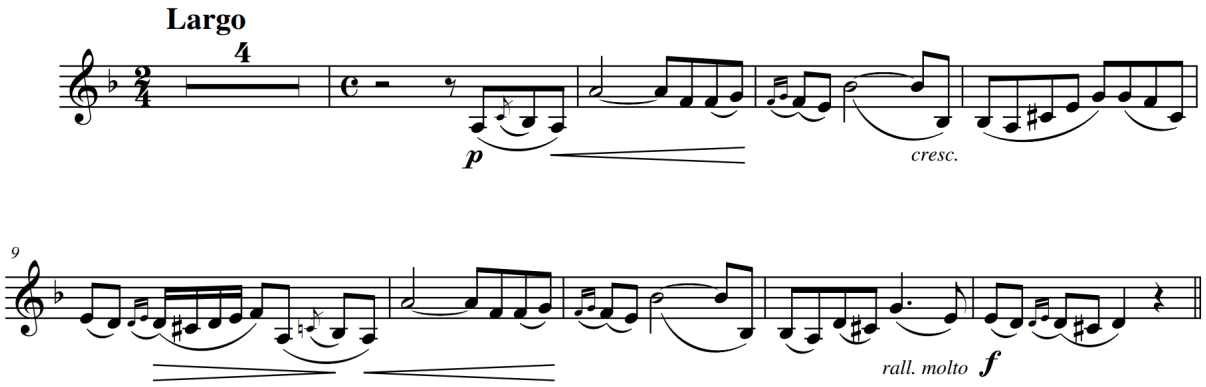


Figure 9: Partition de la Largo de Csárdás (Vittorio Monti) pour violon

On voit bien un bémol à la clé *(Fa majeur)*.

5.6 Tempo et rythme

le temps en musique se définit par le tempo et le rythme.

On définit d'abord **un temps** (en: ***beat***) l'unité de temps en musique.

Le **tempo** du morceau est sa vitesse, il peut être décrit en **BPM** (Beats per minute, fr: *temps par minute*) ou bien par un mot clé en italien: *Largo, Andante, Allegro, Moderato, Lento, Presto, etc* ce qui est moins précis mais bien compris par les musiciens.

Le **rythme** du morceau est sa structure répétitive qui définit une pulsation régulière.

La **mesure** est une structure *rythmique* constitutées de plusieurs *temps* qui se répète périodiquement.

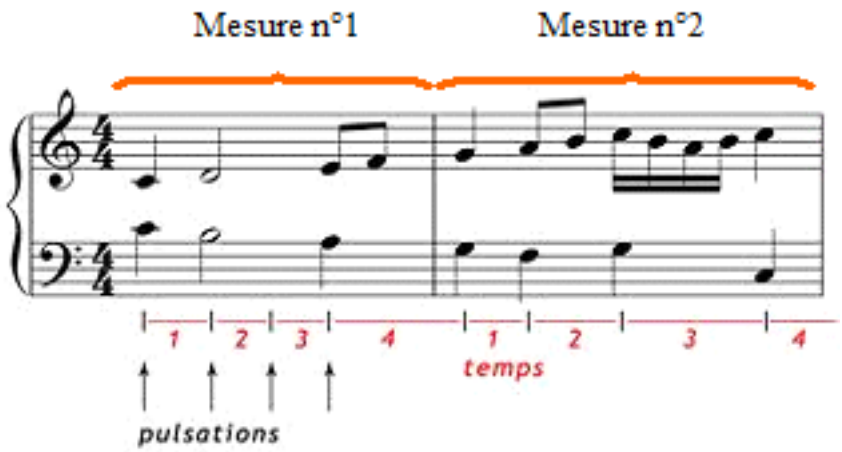


Figure 10: Rythme et tempo

La **métrique** (la division du temps) est le nombre de temps par mesure *et* la valeur d'un temps. Ce sont les 2 chiffres au début d'un morceau, celui en bas indique le temps:

Temps	ronde	blanche	noire	croche	double-croche
Nombre	1	2	4	8	16

Celui en haut indique le nombre de temps (beats) par mesure, par exemple $\frac{3}{4}$ est la métrique qui contient 3 noires par mesure. Les unités les plus utilisées sont les noires et les croches, et rarement la blanche.

5.7 Relation entre le temps et tempo

Il se peut que la métrique soit changé dans un morceau. De plus, le tempo n'est pas forcément constant. Même son évolution au cours du morceau n'est pas toujours linéaire, on peut ralentir au milieu d'un morceau jusqu'à l'arrêt total pour reprendre en plein vitesse d'un seul coup.

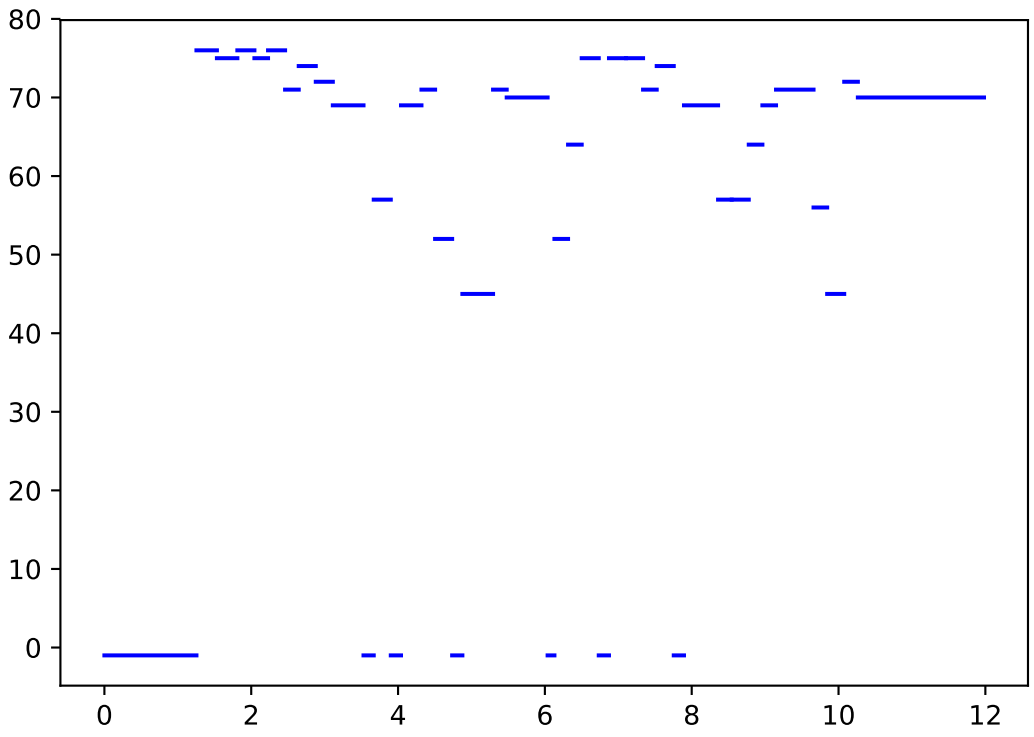
On n'a donc aucune guarantie sur la régularité du tempo.

En revanche, afin de pouvoir tirer des résultats de notre analyse précédantes on a dû prendre des cas particuliers suffisamment réguliers. Par la suite, nous avons définit un **pseudo-PGCD** de durées de notes, et nous avons divisés toutes les durées par ce nombre, et puis en arrondissant le résultats nous avons obtenus des résultats qu'on a écrit en format MIDI.

5.8 Ecriture en format MIDI

On reprend le morceau du piano afin d'obtenir une fonction constante par morceau avec le temps en *beats*, ce qu'on enregistra en format MIDI grâce à la librarie python `midiutil`

```
from muallef import notes
time, pitch = notes.detect(fur_elise.signal, fur_elise.sampleRate)
plt.clf()
plot_step_function(time, pitch)
plt.show()
```




```
notes.extract.extract_midi(time, pitch, "fur_elise.midi")
```

On compare la partition obtenue avec la partition originale:

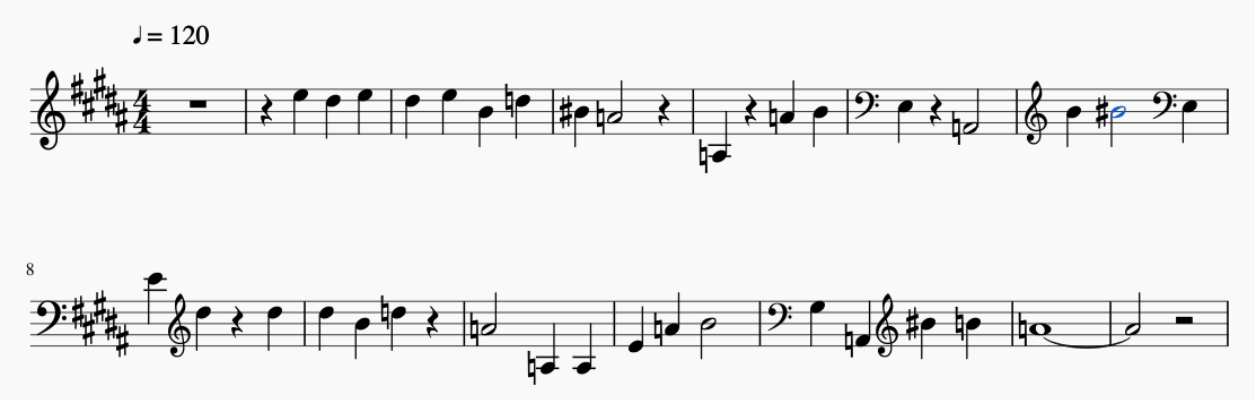


Figure 11: Partition obtenue

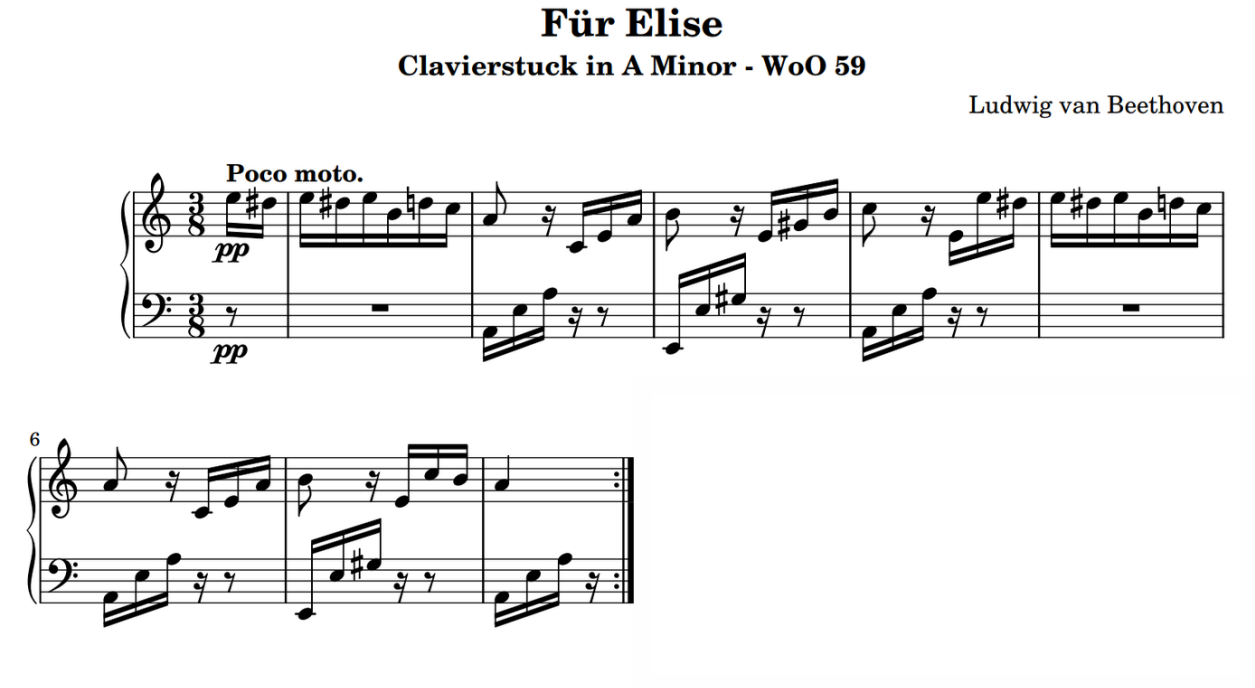


Figure 12: Partition obtenue

On n'a pas obtenu les résultats qu'on espérait avoir, ceci pourra être expliqué par 3 raisons:

- Notre implémentation ne traitent que les son monophones, mais ce morceau contient plusieurs notes simultanées différentes.
- Le morceau est en La mineur (même armure que Do majeur), une gamme sans altérations. Or, dans les 2 lignes qu'on a traités nous avons croisé 3 dièses, d'où l'erreur dans le choix de la gamme.
- Notre traitement du tempo est très élèmentaire, ici on a une métrique composée ce qui est un cas assez délicat pour tel programme.

En revanche, la ségmentation temporelle et la détéction de pitch sont bien réussis.

6 Conclusion

6.1 Résultats

Nous avons réussi à obtenir des résultats certainement intéressants, mais nous n'avons pas pu tout implémenter car nous avons passé beaucoup de temps à la recherche vu la nouveauté du sujet.

Nous aurions aimé pouvoir faire plus de tests avec des exemples variés.

6.2 Développement possible

Nous avons testé les librairies existantes:

- aubio.org
- essentia.upf.edu
- librosa.github.io

Les résultats sont assez satisfaisants. En revanche, ce domaine a du fort potentiel de développement. Notamment le cas de musique pholyphone, et dans la reconnaissance du rythme. D'après ma recherche, le plus intéressant sera de faire intervenir de l'Intelligence Artificielle, en particulier la technologie Long short-term Memory (LSTM). [Wikipedia contributors, 2018]

6.3 Apport personnel

Nous avons proposé ce projet par passion et motivation, ce qui nous a poussé à faire de la recherche scientifique pour la première fois dans notre vie académique.

Nous vous remercions sincèrement pour cette opportunité.

Références

Juan Bello and Mark Sandler. Phase-based note onset detection for music signals. 2003.

Paul Brossier. Automatic annotation of musical audio for interactive applications. 2007.

Chris Duxbury, Juan Bello, Mike Davies, and Mark Sandler. Complex domain onset detection for musical signals. 2003.

Hideki Kawahara and Alain de Cheveigné. Yin, a fundamental frequency estimator for speech and music. 2002.

Paul Masri. Computer modeling of sound for transformation and synthesis of musical signal. 1996.

Carlos Rosão, Ricardo Ribeiro, and David Martins de Matos. Influence of peak selection methods on onset detection. 2012.

Wikipedia contributors. Long short-term memory — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=844880987. [Online; accessed 11-June-2018].

Wikipédia. Chants hourrites — wikipédia, l'encyclopédie libre, 2018. URL http://fr.wikipedia.org/w/index.php?title=Chants_hourrites&oldid=145548895.