Projet Musique Décalée

PCSI/PTSI

Consignes

† Rendu

Le devoir est à faire de manière électronique. Les consignes du rendu sont disponible sur le site web du cours

† Règles de travail

Le travail en groupe et l'entre aide sont encouragés. Toutefois le monde doit rendre son propre travail. Le transfert de code est interdit. Tous codes identiques à renommage et permutations d'instructions près seront sanctionnés par un 0 à la note finale. Pour éviter que cela arrive, la règle à suivre est simple :

- Ne pas copier-coller le code d'un camarade.
- Si un camarade vous explique la solution à partir de son propre code, prendre le temps de digérer l'explication et réécrire plus tard le code de la réponse.

† Code Python

Les consignes suivantes **doivent** être respectées sous peine d'être pénalisé par des **points négatifs**.

Le code doit être clair. - *1 point par fonction.*

- Évitez les lignes à rallonge. S'il faut introduire une variable intermédiaire, faîte le.
- Nommez vos variables avec du sens. Évitez toutefois des noms trop longs. Auquel cas, décrivez le rôle de la variable en commentaire.
- Ne surchargez pas de commentaires. Les commentaires utiles sont :
  - Description du rôle d'une fonction
  - Description du rôle d'une variable
  - Description du rôle d'une boucle complexe.

Respectez la syntaxe Python. - *0 à l'exercice si syntaxe inventée.*

- Gardez la fiche de syntaxe à côté de vous lorsque vous codez
- Vérifiez que ce que vous écrivez respecte la syntaxe.
- Vérifiez que votre code s'exécute avant de l'envoyer.

Pour nommer une fonction, utilisez le nom donné dans l'énoncé - *0 à l'exercice sinon*

Utilisation de `print` ou de `input` interdite - *0 à l'exercice sinon*

L'objectif de ce problème est de générer un fichier de type WAV à partir d'une partition écrite.

I Introduction

Le son est une vibration mécanique d'un fluide qui se propage sous forme d'ondes. On modélisera dans ce problème l'onde par une sinusoïde. Une note de musique correspond à une onde d'une certaine fréquence. Par exemple, le `la`^{3}, note du diapason, correspond à la fréquence 440 Hz. On l'appelle d'ailleurs souvent le `la 440`.

La sinusoïde de fréquence f et d'amplitude A est la fonction

$$\begin{array}{rccc} h: & \mathbb{R} & \to & \mathbb{R} \\ & t & \mapsto & \mathrm{A}\,\sin(2\,\pi\,f\,t) \end{array}$$

Un haut-parleur permet de produire du son à partir d'un signal électrique. Le son est produit grâce aux vibrations d'une membrane. Cette membrane est reliée à un aimant qui est lui-même attiré et repoussé par une bobine en fonction du courant électrique qui la traverse.

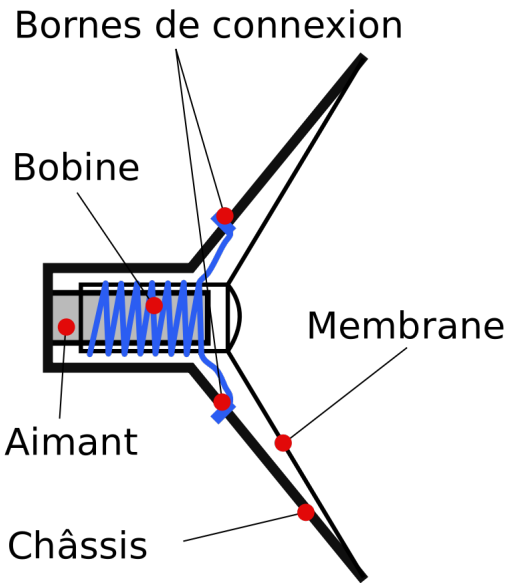

FIGURE 1 – Schéma d'un Haut-Parleur

Le son analogique (signal électrique) correspondant à une note de musique est produit par la carte son de l'ordinateur. Cette dernière est capable de traduire un son numérique (données binaires) en son analogique.

Le signal analogique est un signal continu. Le son numérique consiste en un *échantillonnage* du signal analogique, *i.e* à extraire un nombre fini de valeurs du signal.

Lorsque les valeurs sont prises à intervalles réguliers, le nombre de valeurs extraites en une seconde est appelée *fréquence d'échantillonnage*. On définit aussi *l'amplitude* de l'échantillonnage comme étant l'amplitude de la fonction sinusoïde.

Un fichier `WAV` est un fichier qui contient certaines informations sur la nature du son à produire par la carte son ainsi qu'un échantillonnage du signal sonore à produire.

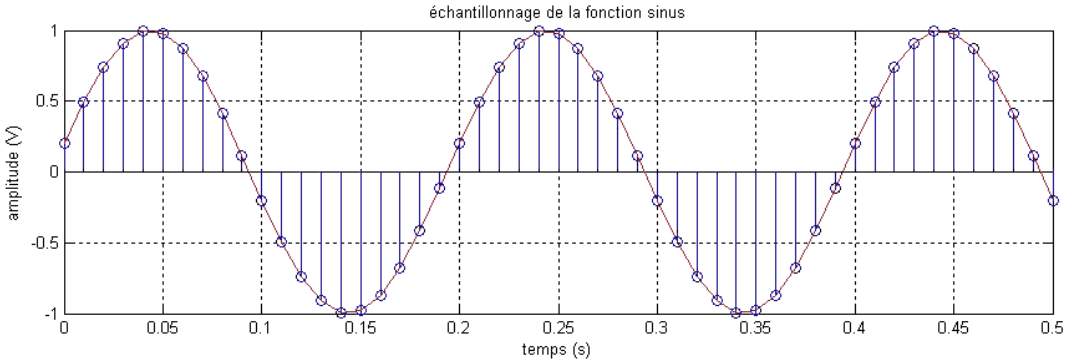

FIGURE 2 – Échantillonnage de la fonction sinus

Le but du projet est donc de produire un échantillonnage du signal sonore correspondant à une partition écrite sous format numérique textuel.

Dans un premier temps, les notes seront représentées par le numéro de la touche qui la produit sur un piano à 88 touches. Par exemple, le `la 440` est produit par la touche 49 du piano.

Si n est le numéro de la touche de piano qui produit une note, la fréquence de cette note s'écrit :

$$2^{\frac{n-49}{12}} \cdot 440$$

QUESTION 1.1. *Fréquence d'une touche de piano*

Implémentez une fonction `freq_touche` qui prend en entrée un entier n et renvoie la fréquence en Hz de la touche n du piano si $0 < n < 89$, 0 sinon.

II Mélodie MIDI depuis une liste

2.1 Outils numériques

Dans cette section, on commencera par implémenter des fonctions permettant de faire des calculs numériques utiles pour l'échantillonnage.

QUESTION 2.1. *Valeur absolue*

Implémentez une fonction `abs` qui prend en entrée un nombre flottant et renvoie sa valeur absolue.

QUESTION 2.2. *Approximation de pi*

On définit la suite réelle, récurrente, suivante qui tend vers π :

$$P_n = \begin{cases} 1 & \text{si } n = 0 \\ P_{n-1} + \frac{(-3)^{-n}}{2n+1} & \text{sinon} \end{cases}$$

Implémentez une fonction `approx_pi` qui prend en entrée un entier n et qui renvoie $12^{\frac{1}{2}} P_n$. Vérifiez que `approx_pi` renvoie la même valeur pour 40, 100, 1000 et 100.000. Comparez les décimales obtenues à celle de π.

On pourra alors définir une variable `pi` dans l'environnement global comme ceci :

— si la question a été réussie et que `approx_pi` (40) est égal à `3.141592653589793` à 10^{-14} près :

```
pi = approx_pi(40)
```

— sinon

```
pi = np.pi
```

Question 2.3. *Modulo deux pi*

Implémentez une fonction `modulo_2pi` qui prend en entrée un flottant x et qui renvoie sa valeur modulo 2π. On rappelle que x modulo 2π est l'unique flottant $y \in]-\pi, \pi]$ tel que $\exists\, n \in \mathbb{N}$, $|x-y| = 2\pi n$. On note alors $y \equiv x\ [2\pi]$.

Question 2.4. *Approximation du sinus*

Pour tout $x \in \mathbb{R}$, on définit la suite réelle suivante qui tend vers $\sin(x)$:

$$s_n^x = \begin{cases} x & \text{si } n = 0 \\ -s_{n-1}^x \cdot \frac{x^2}{2n(2n+1)} & \text{sinon} \end{cases}$$

$$S_n^x = \sum_{i=0}^{n} s_i^x$$

Implémentez une fonction `approx_sin` qui prend en entrée un flottant x, un entier n puis calcule $y \equiv x\ [2\pi]$ et renvoie S_n^y.

Attention la convergence est trop lente si x n'est pas dans $]-\pi, \pi]$, il faut absolument prendre le modulo…

2.2 Échantillonnage

Une partition est donnée dans un premier temps comme étant une liste de couple `(n,d)`, avec `n` le numéro de la touche de piano produisant la note et `d` la durée de la note en *huitième de secondes*.

Les *tuples* en Python se comporte comme les listes, mais sont *immuables* : ils ne peuvent pas être modifiés. Ils se notent avec des parenthèses et les fonctions `append` et `pop`, ainsi que l'affectation à un indice sont impossibles. La syntaxe est décrite sur la fiche de syntaxe.

L'échantillonnage d'une partition est obtenu en concaténant l'échantillonnage de chaque couple. L'échantillonnage d'un couple `(n,d)` correspond à l'échantillonnage de la sinusoïde modélisant l'onde sonore produite par l'appui de la touche `n` pendant $\frac{d}{8}$ secondes. Les valeurs de l'échantillonnage sont donc prises de manière régulière dans l'intervalle $[0, \frac{d}{8}]$.

Par exemple, voici un échantillonnage du `la 440` sur une durée de 2 sec (16 huitièmes de secondes) avec comme fréquence d'échantillonnage `5 Hz` et comme amplitude `2`.

```
[ 0.0 , -1.9696 ... , -0.6840 ... , 1.7320 ... , 1.2855 ... , -1.2855 ... ,
-1.7320 ... , 0.6840 ... , 1.9696 ... , -6.7165 ... e-12 ]
```

Question 2.5. *Discrétisation d'un intervalle*

Implémentez une fonction `discretisation` qui prend en entrée deux flottants $a < b$ et un entier $n \geq 2$ et qui renvoie une *liste* de n points (flottants) de l'intervalle $[a, b]$ à écarts réguliers.

Formellement la sortie est une liste $l = [i_1, i_2, \cdots, i_n]$ telle que :

1. $i_1 = a$
2. $i_n = b$
3. $\exists\, \alpha \in \mathrm{R}^{+*},\ \forall\, 1 \leq k < n,\ i_{k+1} - i_k = \alpha$

Indication : Déterminez α puis pour tout $1 \leq k < n$ la valeur de i_k.

Question 2.6. *Échantillonnage d'une note*

Implémentez une fonction `echantillonnage_note` qui prend en argument la fréquence de la note, sa durée, la fréquence d'échantillonnage et l'amplitude et qui renvoie l'échantillonnage de la note.

Pour se faire, on commencera par définir la liste des points sur lesquels sera fait l'échantillonnage : `t = discretisation(a,b,n)`, avec `a`, `b` et `n` des valeurs à définir vous-même.

Pour calculer le sinus d'une valeur flottante `x`, on pourra utiliser :

— `approx_sin(x,100)`, si `approx_sin(2*pi,100)` est égal à 0 à 10^{-14} près
— `np.sin(x)`, sinon.

Question 2.7. *Échantillonnage d'une partition*

Implémentez une fonction `echantillonnage_partition` qui prend en argument une partition, la fréquence d'échantillonnage et l'amplitude et qui renvoie l'échantillonnage de la partition.

Vous devrez utiliser la fonction `echantillonnage_note` implémentée à la question précédente pour récupérer l'échantillonnage d'une note. Vous pourrez utiliser cette fonction même si vous n'avez pas réussi à l'implémenter.

Question 2.8. *Génération du fichier WAV*

Cette question n'est pas notée.

On peut maintenant utiliser l'échantillonnage que vous avez produit pour écrire un fichier WAV. Dans la console de Spyder vous pouvez ajouter le code suivant pour générer un fichier WAV dans le même dossier que votre fichier Python.

```
freq_e = 44100 # Fréquence d'échantillonnage
amplitude = 2048 # Amplitude de l'échantillonnage
# Changez les notes et durées ci-dessous pour générer d'autres mélodies...
# Do Ré Mi Fa Sol La Si - 2sec chaque
partition = [(40,16),(42,16),(44,16),(45,16),(47,16),(49,16),(51,16)]
e = echantillonnage_partition(partition,freq_e,amplitude)
wavfile.write('octave4_2s.wav',
              rate=freq_e,
              data=np.array(e).astype(np.int16) )
```

Vous pouvez m'envoyer par mail vos plus belles mélodies. Elles seront mise à l'écoute sur le site web, de manière anonyme ou non selon votre préférence.

Question 2.9. *Échantillonnage d'un accord*

Un accord de piano consiste à appuyer sur plusieurs touches en même temps. L'onde sonore produite est la superposition des ondes sonores de chaque touche. Elle est modélisée par la somme des sinusoïdes de chaque note. Par exemple, pour l'accord de `fa`[3] on appuie sur les touches `fa` (`45`), `la` (`49`) et `do` (`52`). On note $f_{\texttt{fa}}$, $f_{\texttt{la}}$ et $f_{\texttt{do}}$ la fréquence de ces trois notes. L'onde sonore de l'accord est modélisée par la fonction :

$$\begin{array}{rccl} h: & \mathbb{R} & \to & \mathbb{R} \\ & t & \mapsto & A\left(\sin(2\pi f_{\texttt{fa}} t) + \sin(2\pi f_{\texttt{la}} t) + \sin(2\pi f_{\texttt{do}} t)\right) \end{array}$$

Implémentez une fonction `echantillonnage_accord` qui prend en entrée une *liste de fréquences,* une durée, une fréquence d'échantillonnage et une amplitude et renvoie l'échantillonnage de l'accord dont les fréquences sont celles de la liste.

Question 2.10. *Échantillonnage d'une partition d'accord*

Une partition d'accord est une liste de couples `(l,d)` ou `l` est une liste $[n_0, n_1, \cdots, n_k]$ de numéro de touche de piano représentant un accord de piano et `d` est la durée de l'accord.

Implémentez ensuite une fonction `echantillonnage_partition_accords` qui prend en entrée une partition, une fréquence d'échantillonnage et une amplitude et qui renvoie l'échantillonnage de la partition.

Vous pouvez générer un fichier WAV à l'aide du code ci-dessous

```
freq_e = 44100
amplitude = 2048
# Vous pouvez changer la partition ci-dessous
# Accord de Fa, La, Accord de Re
partition = [([49,52,45],16), ([49],8), ([42,46,49],16)]
e = echantillonnage_partition_accords(partition,freq_e,amplitude)
wavfile.write('accords.wav',
              rate=freq_e,
              data=np.array(e).astype(np.int16) )
```

III Mélodie MIDI depuis une partition écrite

On veut maintenant écrire la partition, non plus sous forme de liste de couple, mais sous la forme d'un texte dans lequel les notes sont écrites avec la notation grégorienne. Les notes de la gamme tempérée sont notées dans l'ordre `C, C#, D, D#, E, F, F#, G, G#, A, A#, B` qui correspond donc à `Do, Do#, Ré, Ré#, Mi, Fa, Fa#, Sol, Sol#, La, La#, Si`. Pour simplifier la lecture de la partition, on modifie la notation des demi-tons : les dièses sont notés par la version minuscule de la note. Ainsi `Do#` qui se note en grégorien `C#` sera noté `c`.

On apposera derrière le caractère décrivant la note le numéro de l'octave dans laquelle elle se trouve. Attention, la numérotation des octaves diffère selon la notation. Par exemple, le `La#` de l'octave `2` sera noté `a2`. Les octaves du piano à 88 touches vont de 0 à 7. La touche numéro 1 est la note `A0` et la touche 88 est la note `C8`. Les touches produisent ensuite les notes de chaque octave dans l'ordre de la gamme et par octaves croissante :

`A0, a0, B0, C1, c1, ... , A6, a6, B6, C7`

Pour plus de détails, vous pouvez lire la page Wikipédia sur les touches de pianos.

Question 3.1. *Touche correspondant à une note dans l'octave*

En faisant une recherche linéaire dans la chaîne de caractère `"CcDdEFfGgAaB"` implémentez une fonction `touche_octave` qui prend en entrée un caractère et renvoie son numéro entre `0` et `11` dans l'octave, *i.e* dans la chaîne listant les notes de la gamme tempérée. Si le caractère n'est pas une note de la gamme tempérée, la fonction doit renvoyer `-1`.

Question 3.2. *Touche correspondant à une note dans l'octave*

Soit `s` un caractère et `k` un numéro d'octave. On note `t = touche_octave(s)`. La touche correspondant à la note `s` de l'octave `k` est

$$n \begin{cases} -1 & \text{si } t\ -1 \\ t - 8 + k * 12 & \text{sinon} \end{cases}$$

Implémentez une fonction `touche_piano` qui prend en entrée un caractère `s` et un numéro d'octave `k` et renvoie le numéro de la note `s` dans l'octave `k` entre `1` et `88` si c'est une note valide, un nombre strictement plus petit que 1 ou strictement plus grand que 89 sinon.

Question 3.3. *Lecture d'une partition*

Une note s'écrira toujours comme une chaîne de trois caractères, note, octave, durée. Par exemple : `"G62"` représente le `Sol` de l'octave `5` pour une durée de $\frac{1}{4}$ secondes, `"d41"` représente le `Re#` de l'octave `4` pour une durée de $\frac{1}{8}$ secondes.

Une partition est la concaténation des chaînes de caractère représentant chaque notes de la partition : `"C41D41E41F41G41A41B41C51B41A41G41F41E41D41C41"`.

Implémentez une fonction `partition` qui prend en entrée une chaîne de caractère représentant une partition et qui renvoie une partition dans le format utilisé par la fonction `echantillonnage_partition`, c'est-à-dire une liste de couple `(n,d)` avec `n` le numéro de la touche de piano correspondant à la note et `d` la durée de la note.

Une chaîne de caractère `s` qui représente un nombre peut être convertie en ce nombre en utilisant la fonction **`int`** : **`int`**`("12")` donne l'entier `12`.

Question 3.4. *Conversion texte vers WAV*

Cette question n'est pas notée.

Vous pouvez alors générer un fichier WAV à partir d'une partition sous forme de chaîne de caractère avec le code suivant dans la console Spyder :

```
freq_e = 44100
amplitude = 2048
p = partition("C41D41E41F41G41A41B41C51B41A41G41F41E41D41C41") # partition
e = echantillonnage_partition(p,freq_e,amplitude)
wavfile.write('octave4_asc_desc_1s.wav', rate=freq_e,
                      data=np.array(e).astype(np.int16) )
```

Question 3.5. *Conversion fichier vers WAV*

On veut maintenant générer le fichier WAV en lisant la partition dans un fichier de type TXT. La fonction `open` permet d'ouvrir un fichier :

```
f = open(chemin,mode)
```

`chemin` décrit l'emplacement de votre fichier sur l'ordinateur et `mode` permet de préciser le mode d'ouverture de votre fichier (lecture, écriture, ajout, …).

Ainsi, pour ouvrir un fichier `partition.txt` situé dans le dossier `songs`, en mode lecture, on utilisera le code :

```
f = open("songs/partitition.txt",'r')
```

Puis la fonction `read` permet de récupérer le contenu du fichier dans une chaîne de caractère :

```
p = f.read()
```

L'écriture d'un fichier WAV se fait à l'aide de l'instruction suivante :

```
wavfile.write(wave, rate=freq_e, data=np.array(e).astype(np.int16) )
```

où `wave` est le chemin du fichier à écrire, par exemple `"melodie.wav"`, `freq_e` est la fréquence d'échantillonnage et `e` est l'échantillonnage de la mélodie.

Implémentez une fonction `generer_partition` qui prend en entrée une chaîne de caractère, `fichier`, contenant le chemin du fichier à lire, une seconde chaîne de caractère, `wave`, contenant le chemin du fichier à écrire, la fréquence d'échantillonnage et l'amplitude et qui :

- lit la partition contenue dans `fichier`
- écrit la mélodie correspondante dans le fichier `wav`
- renvoie `None`

Par exemple, `generer_partition("songs/weezer.txt","melodie_weezer.wav",44100,2046)` permet de lire la partition `weezer.txt` et de générer la mélodie correspondante `melodie_weezer.wav` avec une fréquence d'échantillonnage de `44100` et une amplitude de `2046`.