Travaux Pratiques - OASIS SI101

**SLI et filtres récursif**

---

## Préliminaires : Lien vers le colab

cliquer ici ou taper le lien ci-dessous.

`https://colab.research.google.com/drive/1fEWFQLNVeEy4RIOnUdecsXoOrUJSx0qq?usp=sharing`

Une fois dans colab, vous pouvez soit le copier et utiliser sur google colab ou bien télécharger le fichier notebook et le lancer sur votre ordinateur.

# 1 Écho et fréquences réduites

## 1.1 Écoute d'un écho

Charger un son dans python à l'aide de la fonction (exemple dans le colab)

`lire_fichier_son`

La commande **joue_son** permet de jouer un son.

On peut modéliser un écho par le fait que le son s'ajoute à lui même avec un certain retard, si $u$ représente le signal sonore original, le signal reçu sera du type

$$v_n = u_n + 0.8u_{n-t_1}$$

Le temps $t_1$ sera le temps de décalage.

Ceci signifie que le son rebondit sur un obstacle, perd 20% de son amplitude et arrive retardé de $t_1$. On constate que cette modélisation revient à réaliser la convolution du son $u$ avec une réponse impulsionnelle $h$ définie par $h_0 = 1,\ h_{t_1} = 0.8$ (tous les autres termes de $h$ sont nuls).

Écouter un son d'origine, puis utiliser la cellule adéquate pour écouter un écho avec 0,1 seconde puis 0,5 et 0,8 secondes.

**Q**.1 Dans le code fourni on utilise effectivement la convolution, donner la complexité du calcul pour un écho de 0,1 secondes et une fréquence d'échantillonnage de 44000 ? Pouvez-vous donner un algorithme plus efficace (dont la complexité ne dépend pas de $t_1$) ?

## 1.2 Fréquence en Hz et réduite

Générer une onde de fréquence 0,01 et longue de 100000 échantillons ($n \mapsto e^{2i\pi\nu n}$, $\nu$ est la fréquence). Écoutez sa partie réelle(`np.real(u)`) par

```
joue_son(real(u),44100) # Le son est joué à 44100 échantillons par seconde
joue_son(real(u),22050)# le son est joué à 22050 échantillons par seconde
```

**Q**.2 Quelles fréquences en Hz (1Hz=1/seconde) percevez-vous dans les deux cas ?

On dit que la fréquence en Hz est la fréquence réelle (ce qui nécessite de connaître la fréquence d'échantillonnage) alors que la fréquence 0,01 est la fréquence réduite (qui est toujours, pour un signal défini sur $\mathbb{Z}$, entre $-1/2$ et $1/2$).

## 2 Filtre rejecteur

Dans la suite on utilise la commande python lfilter (qui doit être définie par lfilter=scipy.signal.lfilter).

**Q**.3 Après avoir consulté l'aide par help(lfilter), dire comment utiliser lfilter pour effectuer une convolution d'un signal contre un RIF (réponse impulsionnelle finie).

Dans la suite nous allons voir comment utiliser lfilter pour effacer une fréquence parasite dans un son. Pour cela nous créons une version polluée de la variable **y** :

```
n=np.arange(len(y)); #le temps discret que dure le signal y
f0=1261;
yb=y+0.1*cos(2*pi*f0/Fe*n); #on ajoute une onde parasite
```

Écouter le son **yb** à la fréquence **Fe**.

**Q**.4 Quelle est la fréquence en Hz et fréquence réduite de l'onde parasite ?

**Q**.5 Donner la TZ de la RI du filtrage qu'effectue la fonction rejette1 ?

Écouter le résultat de ce filtrage sur le signal **yb**.

**Q**.6 Commentaire ?

**Q**.7 Donner la TZ de la RI du filtrage qu'effectue la fonction rejette2 ?

Écouter le résultat sur le signal **yb**.

**Q**.8 Pour une valeur de **rho** proche de 1, placer les zéros et les pôles des TZ précédentes dans le plan complexe. Utiliser ces positions pour tracer (approximativement) le module des deux TFtD correspondantes (sur l'intervalle [-1/2,1/2[). Conclure quand à la qualité comparée des deux filtrages.

**Q**.9 Pourquoi **rho** (paramètre de rejette2) doit-il toujours être strictement plus petit que 1 ?

**Q**.10 En transmettant à la fonction rejette2 une fréquence **f0=1267** et un paramètre **rho** très très proche de 1, conclure au compromis nécessaire dans le choix de **rho**.