Département Génie électrique et informatique
Génie des systèmes embarqués et informatique industrielle

RAPPORT DE TP2
Traitement de la parole

Objectifs :

Analyse spectrale du signal de la parole
Analyse spectrale temps-court
Analyse LPC du signal de la parole par la méthode des treillis

Réalisé par :

ISMAILI El Yazid

MOUATASSEM Soumaya

Encadré par :

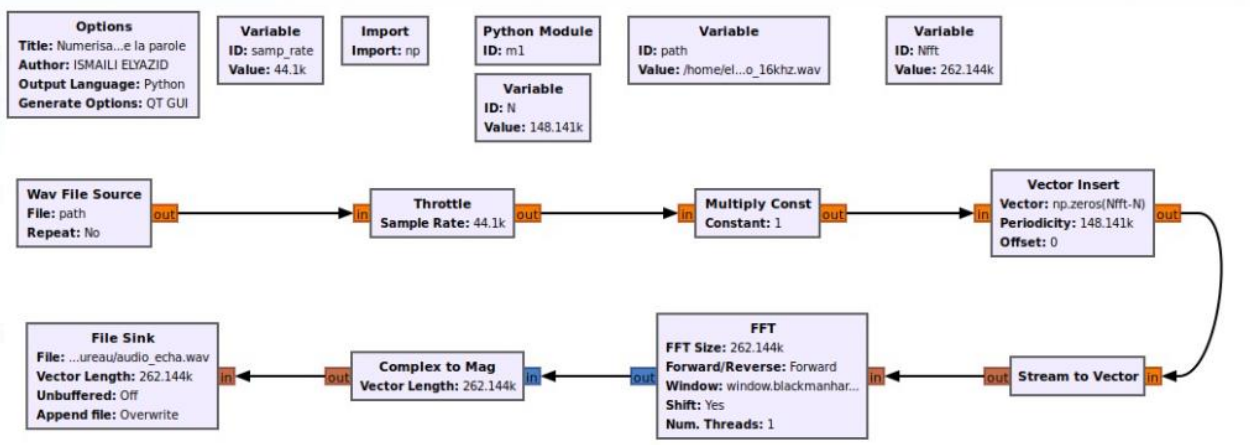
Pr. BELKEBIR Hicham

Année universitaire
2021-2022

Phase 1 : Analyse spectrale du signal de la parole

- Dans cette partie on vise à calculer le spectre entier de la séquence audio enregistrée sur l'ordinateur et l'afficher en utilisant un script externe à GNU Radio Companion. Pour se faire, on utilise les blocs suivants :

Nous avons besoin dans ce cas de deux fonctions, la première pour calculer la taille de mon fichier audio et donc savoir combien d'échantillons y'en a , et la deuxième pour savoir le nombre de Zeros que la fonction d'interpolation doit implémenter .

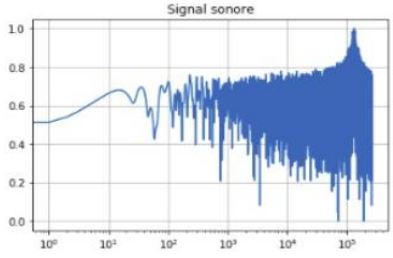


On a implémenté les fonctions suivantes dans le module Python :

```python
import numpy as np

def length_stream(Path='/home/elyazid/Bureau/audio_16khz.wav'):
        return len(np.fromfile(Path))

def nextpow2(N):
        n=0
        while 2**n <N :
                n=n+1
        return 2**n
```

- <u>La visualisation des résultats à partir le script Python</u>

```
Entrée [25]:
import numpy as np
from math import log10
import matplotlib.pyplot as plt

Entrée [30]:
file = np.fromfile("/home/elyazid/Bureau/audio_echa.wav", dtype=np.float32)
arr = 20*np.log10(file)
plt.title(r"Signal sonore")
arr=((arr-min(arr))/(max(arr)-min(arr)))
plt.plot(arr)
print(max(file))
plt.xscale("log")
plt.grid()
plt.show()

866.39026
```

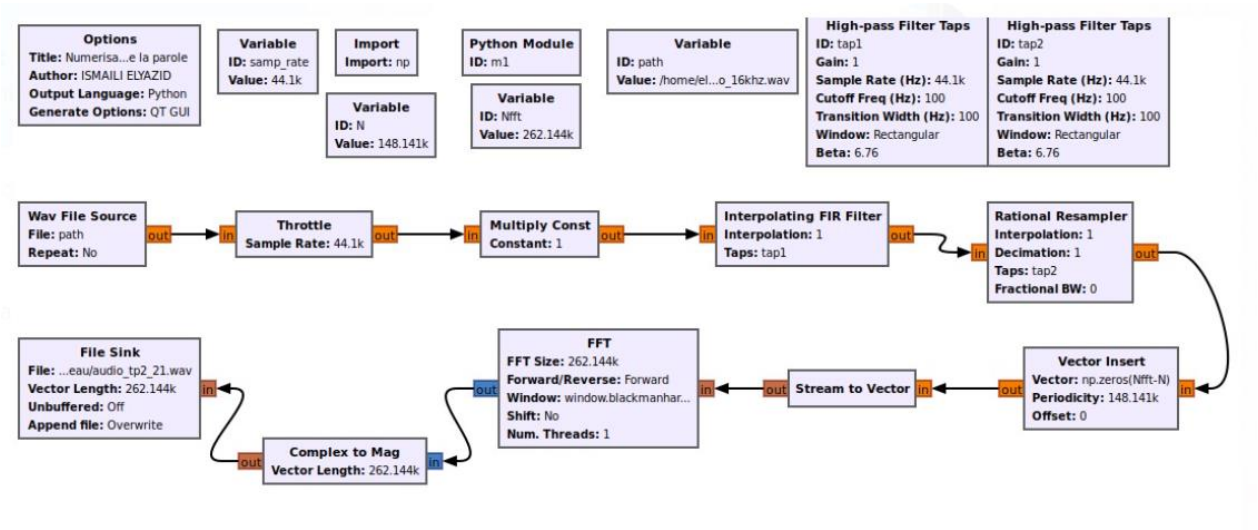


On ne visualise que la partie positive (droite) du signal , on remarque qu'on ne peut pas extraire les composants du signal ni ses caractéristique , donc on doit opter pour la STFT au lieu de la TFT .

Phase 2 : Analyse Spectrale temps- court

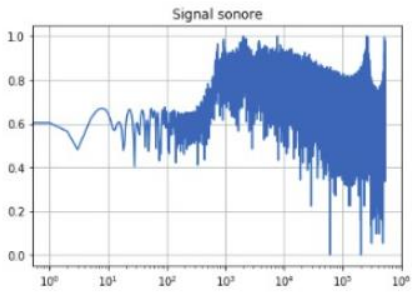
Comme on a vu dans la partie précédente, l'analyse spectrale temps-court permet de multiplier le signal par une fenêtre, puis on va transférer le flux a un vecteur de puissance de pour calculer son spectre.

- **Fenêtre rectangulaire :**

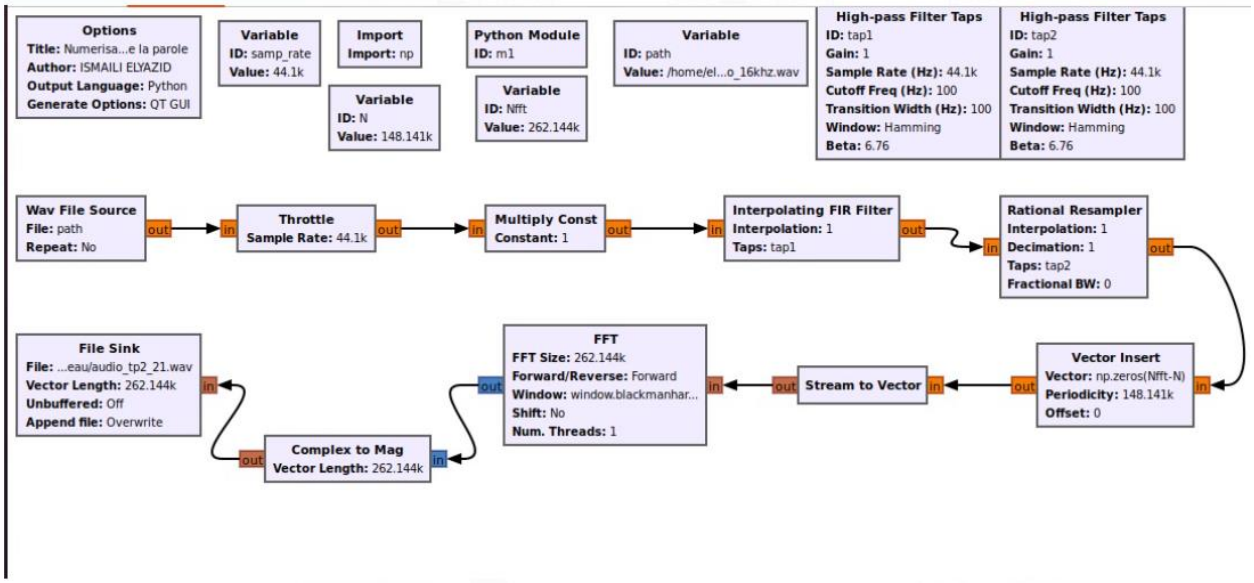


Visualisation des résultats à partir du script python :

```
Entrée [25]: import numpy as np
             from math import log10
             import matplotlib.pyplot as plt

Entrée [34]: file = np.fromfile("/home/elyazid/Bureau/audio_tp2_21.wav", dtype=np.float32)
             arr = 20*np.log10(file)
             plt.title(r"Signal sonore")
             arr=((arr-min(arr))/(max(arr)-min(arr)))
             plt.plot(arr)
             print(max(file))
             plt.xscale("log")
             plt.grid()
             plt.show()

             754.5358
```



- **Fenêtre de hamming :**

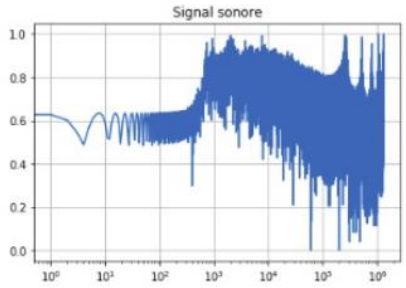


Visualisation des résultats à partir du script python :

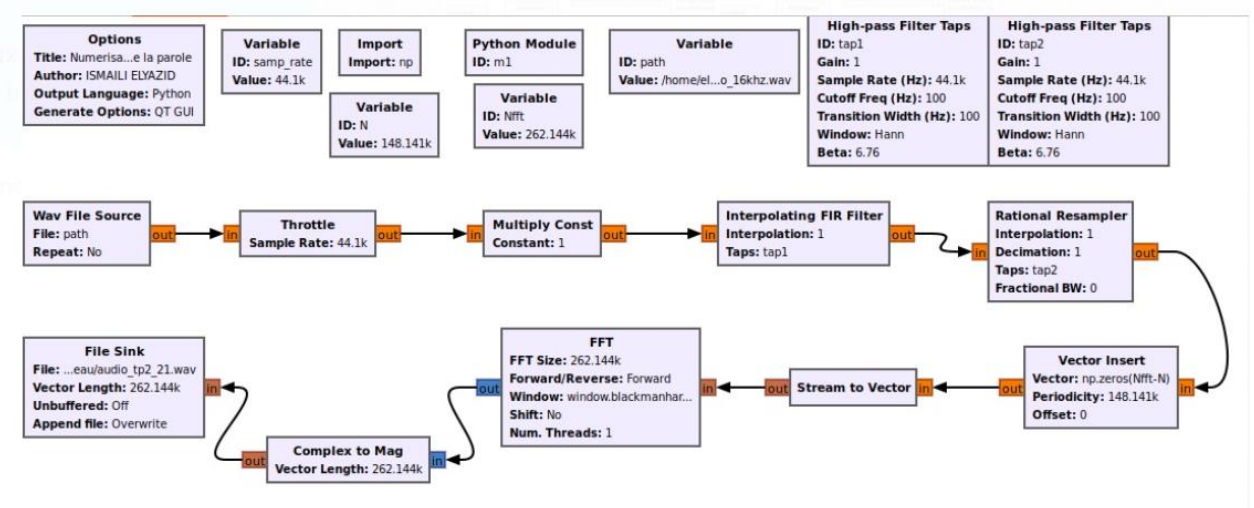
```
Entrée [25]: import numpy as np
             from math import log10
             import matplotlib.pyplot as plt
```

```
Entrée [37]: file = np.fromfile("/home/elyazid/Bureau/audio_tp2_21.wav", dtype=np.float32)
             arr = 20*np.log10(file)
             plt.title(r"Signal sonore")
             arr=((arr-min(arr))/(max(arr)-min(arr)))
             plt.plot(arr)
             print(max(file))
             plt.xscale("log")
             plt.grid()
             plt.show()
```

```
827.8765
```



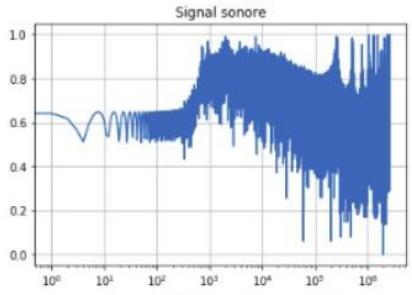
- **Fenêtre de hanning :**



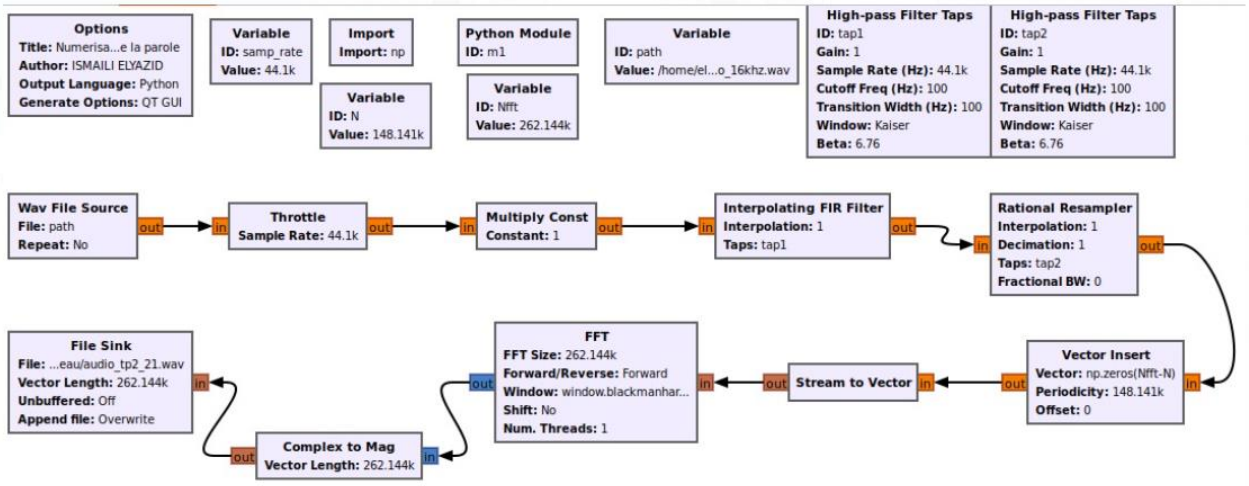
```
Entrée [25]: import numpy as np
             from math import log10
             import matplotlib.pyplot as plt
```

```
Entrée [38]: file = np.fromfile("/home/elyazid/Bureau/audio_tp2_21.wav", dtype=np.float32)
             arr = 20*np.log10(file)
             plt.title(r"Signal sonore")
             arr=((arr-min(arr))/(max(arr)-min(arr)))
             plt.plot(arr)
             print(max(file))
             plt.xscale("log")
             plt.grid()
             plt.show()
```

```
863.06024
```



- **Fenêtre de Kaizer :**

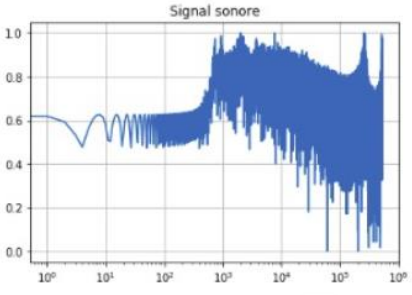


Visualisation des résultats à partir du script python :

```
Entrée [25]: import numpy as np
             from math import log10
             import matplotlib.pyplot as plt
```

```
Entrée [28]: file = np.fromfile("/home/elyazid/Bureau/audio_tp2_2.wav", dtype=np.float32)
             arr = 20*np.log10(file)
             plt.title(r"Signal sonore")
             arr=((arr-min(arr))/(max(arr)-min(arr)))
             plt.plot(arr)
             print(max(file))
             plt.xscale("log")
             plt.grid()
             plt.show()
```

```
762.4483
```



C - Maintenant, on fixe la fenêtre sur le type de Hamming et on varie le temps entre ces valeurs :

$\Delta t :$	$10\ [ms]$	$20\ [ms]$	$30\ [ms]$	$40\ [ms]$

Entrée [25]:
```
import numpy as np
from math import log10
import matplotlib.pyplot as plt
```

Entrée [56]:
```
file1 = np.fromfile("/home/elyazid/Bureau/audio_tp2_21.wav", dtype=np.float32)
file2 = np.fromfile("/home/elyazid/Bureau/audio_tp2_2.wav", dtype=np.float32)
file3 = np.fromfile("/home/elyazid/Bureau/audio_tp2_22.wav", dtype=np.float32)
file4 = np.fromfile("/home/elyazid/Bureau/audio_tp2_23.wav", dtype=np.float32)

arr1 = 20*np.log10(file1)
arr2 = 20*np.log10(file2)
arr3 = 20*np.log10(file3)
arr4 = 20*np.log10(file4)

plt.title(r"Signal sonore")
plt.figure(figsize=(20,10), dpi=80)
arr1=((arr1-min(arr1))/(max(arr1)-min(arr1)))
arr2=((arr2-min(arr2))/(max(arr2)-min(arr2)))
arr3=((arr3-min(arr3))/(max(arr3)-min(arr3)))
arr4=((arr4-min(arr4))/(max(arr4)-min(arr4)))
plt.xscale("log")
plt.subplot(221)
fig1=plt.plot(arr1)
plt.subplot(222)
plt.plot(arr2)
plt.subplot(223)
plt.plot(arr3)
plt.subplot(224)
plt.plot(arr4)
plt.grid()
plt.show()
```

On Obtient les résultats suivants :