

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Nom : LEVEQUE
Prénom : Clément
Numéro de candidat : 29574

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

1 Présentation du sujet

2 Plan du calcul

3 Application

4 Conclusion

5 Annexe

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

1 Présentation du sujet

- Principe
- Etude

2 Plan du calcul

3 Application

4 Conclusion

5 Annexe

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

- But :
 - Recenser les espèces d'oiseaux dans une ville
- Processus :
 - Enregistrements
 - Extractions des informations utiles
 - Reconnaissance

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Coefficients de fréquence Mel

- Reconnaissance de parole
- Reconnaissance de son
- Dépend du temps

1 Présentation du sujet

2 Plan du calcul

- Acquisition et pré-réglages
- Séparation
- Fourier
- Filtrage
- Logarithme et cosinus

3 Application

4 Conclusion

5 Annexe

2 - Plan du calcul

a - Acquisition et pré-réglages

7/41

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

■ Acquisition

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

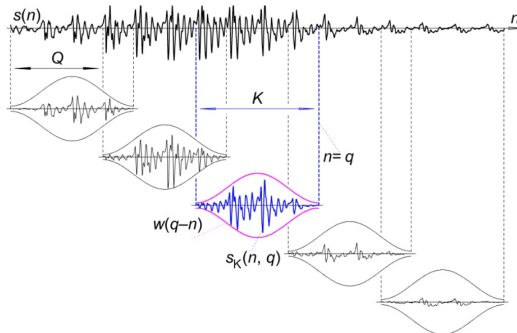
■ Acquisition

■ Centrage :

$$f_n(x) = f(x) - m$$

m moyenne, f fonction du signal et f_n fonction centrée

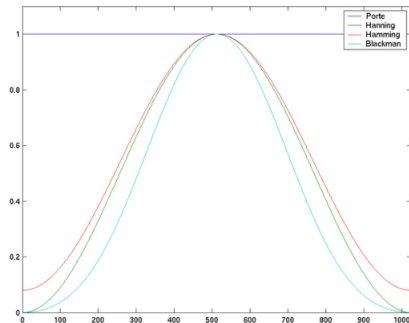
- Acquisition
- Centrage :
$$f_n(n) = f(n) - m$$
 m moyenne, f fonction du signal et f_n fonction centrée
- Pré-accentuation :
$$f_a(n) = f(n) - cf(n - 1)$$
 $c \in [0, 9; 1]$, f fonction du signal et f_a fonction accentuée



<https://yannick-1.gitbook.io/reconnaissance-automatique-de-la-parole/chapter1>

- s signal, Q longueur d'une trame, K taille d'un échantillon, q l'échantillon, s_K signal extrait, w fonction de fenêtrage
- $s_K(n, q) = s(n)w(q - n)$

Fonctions de fenêtrage



<https://fr.wikipedia.org/wiki/Fen>

Fenêtre de Hamming :

$$w(n) = 0.54 - 0.46 \cos(2\pi \frac{n-1}{K-1})$$

Décomposition en série de Fourier finie

Pour tous les échantillons :

- N taille de la liste, $W = \exp(-\frac{2i\pi}{N})$, $(X_k)_{k \in \llbracket 0; N-1 \rrbracket}$ liste d'entrée et $(A_r)_{r \in \llbracket 0; N-1 \rrbracket}$ liste de sortie
- $\forall r \in \llbracket 0; N-1 \rrbracket, A_r = \sum_{k=0}^{N-1} X_k W^{rk}$

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

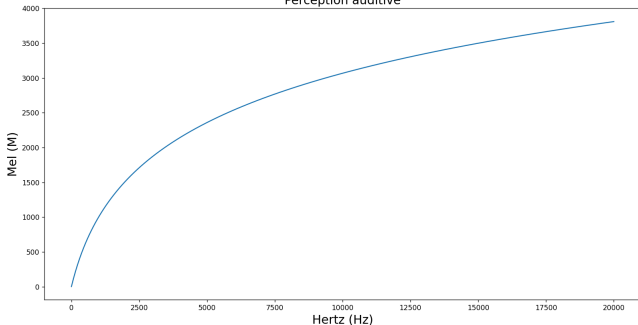
Spectrogramme

Pour tous les échantillons :

- $\forall r \in \llbracket 0; N-1 \rrbracket, A_r^s = \frac{|A_r|^2}{N}$
- $(A_r^s)_{r \in \llbracket 0; N-1 \rrbracket}$, liste des points du spectrogramme

Domaine de Mel

Perception auditive



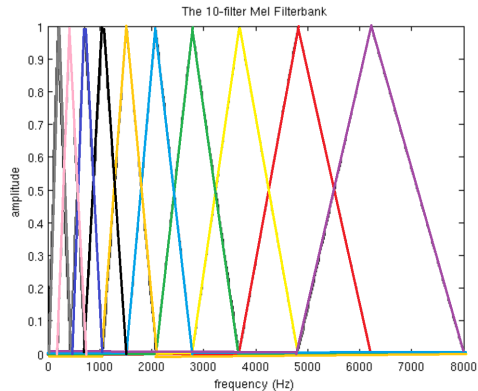
- $F_{mel}(F_{hz}) = 1125 \log(1 + \frac{F_{hz}}{700})$
- $F_{hz}(F_{mel}) = 700(\exp(\frac{F_{mel}}{1125}) - 1)$
- F_{hz} la fréquence en hertz et F_{mel} la fréquence en mel

Filtres

- F_{max} fréquence maximale et F_{min} fréquence minimale N nombre de filtres
- $M_{max} = F_{mel}(F_{max})$ et $M_{min} = F_{mel}(F_{min})$
- $(M_n)_{n \in \llbracket 0; N+1 \rrbracket}$ liste de $N + 2$ points équitablement répartis entre M_{max} et M_{min}
- $(H_n)_{n \in \llbracket 0; N+1 \rrbracket}$ liste telle que
 $\forall n \in \llbracket 0; N + 1 \rrbracket, H_n = F_{hz}(M_n)$
- $\forall m \in \llbracket 0; N - 1 \rrbracket, \phi_m(k) =$

$$\begin{cases} 0 & \text{si } k < H_{m-1} \\ \frac{k - H_{m-1}}{H_m - H_{m-1}} & \text{si } H_{m-1} \leq k \leq H_m \\ \frac{H_{m+1} - k}{H_{m+1} - H_m} & \text{si } H_m \leq k \leq H_{m+1} \\ 0 & \text{si } k > H_{m+1} \end{cases}$$

Filtres



<http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients->

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Passage au logarithme

- N taille de la liste, $(x_n)_{n \in \llbracket 0; N-1 \rrbracket}$ liste d'entrée et $(X_k)_{k \in \llbracket 0; N-1 \rrbracket}$ liste de sortie
- $\forall k \in \llbracket 0; N-1 \rrbracket, X_k = \log(x_n)$

Transformée en cosinus

- N taille de la liste, $(x_n)_{n \in \llbracket 0; N-1 \rrbracket}$ liste d'entrée et $(X_k)_{k \in \llbracket 0; N-1 \rrbracket}$ liste de sortie
- $\forall k \in \llbracket 0; N-1 \rrbracket, X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{2N}(2k+1)n\right)$

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

1 Présentation du sujet

2 Plan du calcul

3 Application

- Problème 1
- Problème 2
- Filtrage
- Problème 3
- Coefficients

4 Conclusion

5 Annexe

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

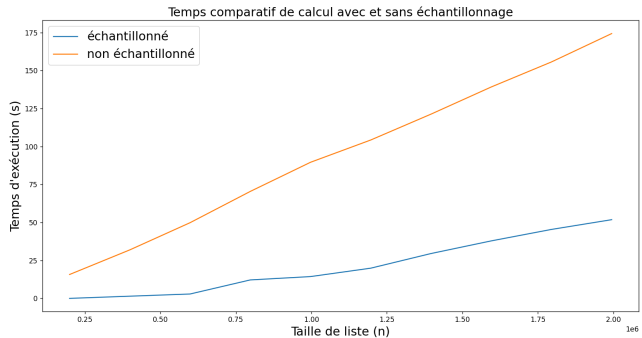
Démonstration

Code

Temps d'exécution

Temps d'exécution

- Solution :
 - Echantillonnage
 - Période



Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Décomposition en série de Fourier finie

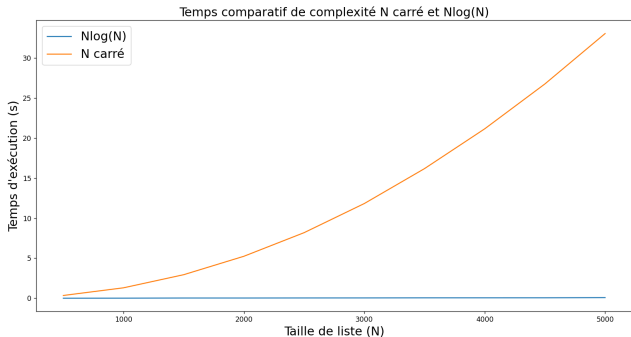
- Complexité $\mathcal{O}(N^2)$

Décomposition en série de Fourier finie

- Complexité $\mathcal{O}(N^2)$
- Solution :
 - Algorithme de Transformée de Fourier Rapide
 - Complexité $\mathcal{O}(N \log(N))$

Décomposition en série de Fourier finie

- Complexité $\mathcal{O}(N^2)$
- Solution :
 - Algorithme de Transformée de Fourier Rapide
 - Complexité $\mathcal{O}(N \log(N))$



Transformée de Fourier Rapide

- N taille de la liste, $W = \exp(-\frac{2i\pi}{N})$, $(X_k)_{k \in \llbracket 0; N-1 \rrbracket}$ liste d'entrée et $(A_r)_{r \in \llbracket 0; N-1 \rrbracket}$ liste de sortie
- $\forall r \in \llbracket 0; N-1 \rrbracket, A_r = \sum_{k=0}^{N-1} X_k W^{rk}$
- $N = 2^p, p \in \mathbb{N}$

Transformée de Fourier Rapide

- Soit $r \in \llbracket 0; N - 1 \rrbracket$, $A_r = \sum_{k=0}^{N-1} X_k W^{rk}$
- Diviser $p - 1$ fois par 2
- Devant chaque terme d'indice impair :
 $W^{r2^x}, x \in \llbracket 0; p - 2 \rrbracket$ avec $W = \exp(-\frac{2i\pi}{N})$
- $\forall r \geq \frac{N}{2} = 2^{p-1}$ on pose $r = 2^{p-1} + j$
- $W^{2^x r} = W^{2^x j}$

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

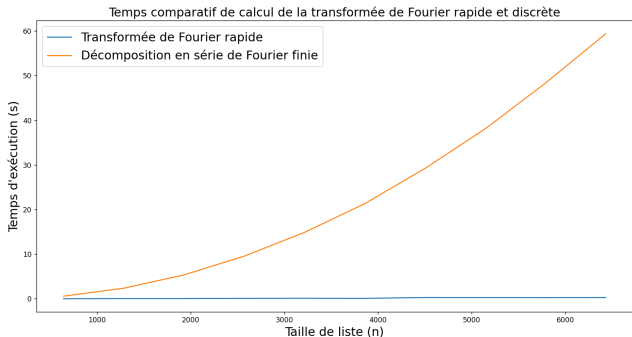
Transformée de Fourier Rapide

Résumé :

- Diviser le problème
- Calcul de la moitié des termes pour chaque sous-problèmes
- Remonter le problème avec les résultats des sous-problèmes précédents

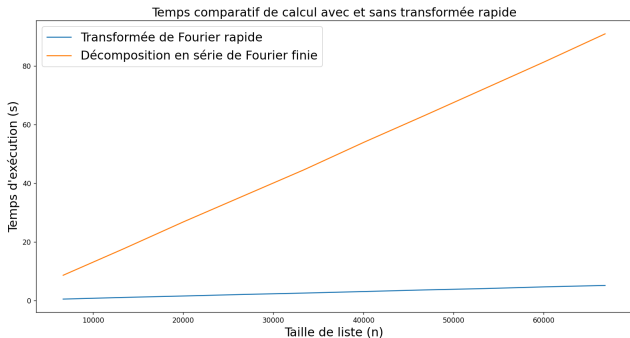
Transformée de Fourier Rapide

Efficacité de l'algorithme :



Transformée de Fourier Rapide

Efficacité du programme complet :



Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

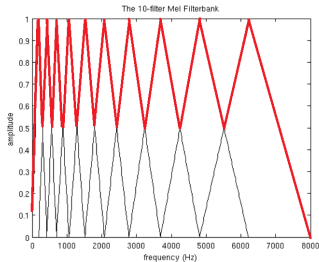
Code

Transformée de Fourier Rapide

Causes de différences :

- Signaux séparés ou non
- Calculs différents

■ Création des filtres



■ On prend les valeurs maximales pour créer ϕ

■ Filtrage : $f_{\text{filtre}} = f * \phi$

Transformée en cosinus discrète

- Complexité $\mathcal{O}(N^2)$
- Solution :
 - Adaptation de l'algorithme de Transformée de Fourier Rapide à la Transformée en cosinus
 - Complexité $\mathcal{O}(N \log(N))$

Transformée en cosinus rapide

- N taille de la liste, $(x_n)_{n \in \llbracket 0; N-1 \rrbracket}$ liste d'entrée et $(X_k)_{k \in \llbracket 0; N-1 \rrbracket}$ liste de sortie
- $\forall k \in \llbracket 0; N-1 \rrbracket, X_k = \sum_{n=0}^{N-1} x_n \cos(\frac{\pi}{2N}(2k+1)n)$
- $N = 2^p, p \in \mathbb{N}$

Transformée en cosinus rapide

- Même principe que la Transformée de Fourier rapide

- Soit $k \in \llbracket 0; N-1 \rrbracket$, $X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{2N}(2k+1)n\right)$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos\left(\frac{\pi(2k+1)2n}{2N}\right) + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(2n+1)}{2N}\right)$$

- $2 \cos\left(\frac{(2k+1)\pi}{2N}\right) \cos\left(\frac{(2k+1)\pi(2n+1)}{2N}\right) =$
 $\cos\left(\frac{2n(2k+1)\pi}{2N}\right) + \cos\left(\frac{2(2k+1)(n+1)\pi}{2N}\right)$

Transformée en cosinus rapide

D'où :

$$2 \cos\left(\frac{\pi(2k+1)}{2N}\right) \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(2n+1)}{2N}\right) =$$

$$\sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)n}{N}\right) + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(n+1)}{N}\right)$$

Reconnaissance
de chant
d'oiseauPrésentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Transformée en cosinus rapide

C'est-à-dire :

$$\sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(2n+1)}{2N}\right) =$$
$$\frac{1}{2 \cos\left(\frac{\pi(2k+1)}{2N}\right)} \sum_{n=0}^{\frac{N}{2}-1} (x_{2n+1} + x_{2n-1}) \cos\left(\frac{\pi(2k+1)n}{N}\right)$$

Transformée en cosinus discrète

Après calculs :

$$\blacksquare \forall k \in \llbracket 0; \frac{N}{2} - 1 \rrbracket, X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos\left(\frac{\pi(2k+1)n}{N}\right) +$$

$$\frac{1}{2 \cos\left(\frac{\pi(2k+1)}{2N}\right)} \sum_{n=0}^{\frac{N}{2}-1} (x_{2n+1} + x_{2n-1}) \cos\left(\frac{\pi(2k+1)n}{N}\right)$$

$$\blacksquare \forall k \in \llbracket 0; \frac{N}{2} - 1 \rrbracket, X_{n-1-k} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos\left(\frac{\pi(2k+1)n}{N}\right) -$$

$$\frac{1}{2 \cos\left(\frac{\pi(2k+1)}{2N}\right)} \sum_{n=0}^{\frac{N}{2}-1} (x_{2n+1} + x_{2n-1}) \cos\left(\frac{\pi(2k+1)n}{N}\right)$$

Transformée en cosinus discrète

Résumé :

- Diviser le problème
- Calcul de la moitié des termes pour chaque sous-problèmes
- Remonter le problème avec les résultats des sous-problèmes précédents

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

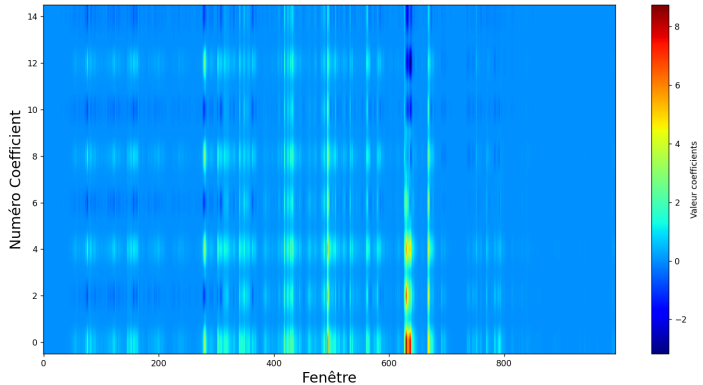
Coefficients

Conclusion

Annexe

Démonstration

Code



Coefficients pour un enregistrement de Pigeon, avec 5 points
par période

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

1 Présentation du sujet

2 Plan du calcul

3 Application

4 Conclusion

5 Annexe

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Objectifs :

- Calcul des coefficients
- Implémentation dans un réseau de neurones

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

1 Présentation du sujet

2 Plan du calcul

3 Application

4 Conclusion

5 Annexe

- Démonstration
- Code

Complexité algorithme de Transformée de Fourier rapide

On pose $C(N)$ la complexité en temps de calcul de la décomposition en série de Fourier discrète, avec $N = 2^p$ la taille de la liste

Si $N \neq 0$ alors : $C(N) = 2C(\frac{N}{2}) + N = 2C(2^{p-1}) + 2^p$

D'où : $\frac{C(2^p)}{2^p} = \frac{C(2^{p-1})}{2^{p-1}} + 1 = p + 1$

C'est-à-dire : $C(2^p) = 2^p(p + 1)$

Or : $\frac{p + 1}{\log(2^p)} = \frac{1}{\log(2)} + \frac{1}{p \log(2)} \xrightarrow{p \rightarrow +\infty} \frac{1}{\log(2)}$

Donc : $C(2^p) = 2^p \mathcal{O}(\log(2^p))$

D'où

$$C(N) = \mathcal{O}(N \log(N))$$

Diviser pour mieux régner

Soit $r \in \llbracket 0; N - 1 \rrbracket$, $A_r = \sum_{k=0}^{N-1} X_k W^{rk}$

$$A_r = \sum_{k=0}^{2^{p-1}-1} X_{2k} W^{2rk} + \sum_{k=0}^{2^{p-1}-1} X_{2k+1} W^{r(2k+1)}$$

$$A_r = \sum_{k=0}^{2^{p-1}-1} X_{2k} W^{2rk} + W^r \sum_{k=0}^{2^{p-1}-1} X_{2k+1} W^{2rk}$$

$$A_r = \sum_{k=0}^{2^{p-2}-1} X_{2(2k)} W^{4rk} + \sum_{k=0}^{2^{p-2}-1} X_{2(2k+1)} W^{2r(2k+1)} +$$

$$W^r \left(\sum_{k=0}^{2^{p-2}-1} X_{2(2k)+1} W^{4rk} + \sum_{k=0}^{2^{p-2}-1} X_{2(2k+1)+1} W^{2r(2k+1)} \right)$$

Reconnaissance
de chant
d'oiseauPrésentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

Calcul de la moitié des facteurs

$$W^{r2^x} = \exp\left(-2i\pi \frac{2^x(2^{p-1}+j)}{2^p}\right)$$

$$W^{r2^x} = \exp(-i\pi 2^x) \exp\left(-2i\pi \frac{2^x j}{2^p}\right)$$

$$\text{Or } \exp(-i\pi 2^x) = 1$$

$$\text{Donc } W^{2^x r} = W^{2^x j}$$

Diviser pour mieux régner

- On pose $K = 2k + 1$
- $2 \cos\left(\frac{K\pi}{2N}\right) \cos\left(\frac{K\pi(2n+1)}{2N}\right)$

$$= 2 \cos\left(\frac{K\pi}{2N}\right) \left(\cos\left(\frac{nK\pi}{N}\right) \cos\left(\frac{K\pi}{2N}\right) - \sin\left(\frac{nK\pi}{N}\right) \sin\left(\frac{K\pi}{2N}\right) \right)$$

$$= 2 \cos\left(\frac{K\pi}{2N}\right)^2 \cos\left(\frac{nK\pi}{N}\right) - 2 \cos\left(\frac{K\pi}{2N}\right) \sin\left(\frac{nK\pi}{N}\right) \sin\left(\frac{K\pi}{2N}\right)$$
- $\cos\left(\frac{2nK\pi}{2N}\right) + \cos\left(\frac{2K(n+1)\pi}{2N}\right)$

$$= \cos\left(\frac{nK\pi}{N}\right) + \cos\left(\frac{2K\pi}{2N}\right) \cos\left(\frac{nK\pi}{N}\right) - \sin\left(\frac{2K\pi}{2N}\right) \sin\left(\frac{nK\pi}{N}\right)$$

$$= \cos\left(\frac{nK\pi}{N}\right) + \left(\cos\left(\frac{K\pi}{2N}\right)^2 - \sin\left(\frac{K\pi}{2N}\right)^2 \right) \cos\left(\frac{nK\pi}{N}\right) -$$

$$2 \cos\left(\frac{K\pi}{2N}\right) \sin\left(\frac{nK\pi}{N}\right) \sin\left(\frac{K\pi}{2N}\right)$$

$$= 2 \cos\left(\frac{K\pi}{2N}\right)^2 \cos\left(\frac{nK\pi}{N}\right) - 2 \cos\left(\frac{K\pi}{2N}\right) \sin\left(\frac{nK\pi}{N}\right) \sin\left(\frac{K\pi}{2N}\right)$$

Addition des deux sommes

- $m = n + 1$:
$$\sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(n+1)}{N}\right) = \sum_{m=1}^{\frac{N}{2}} x_{2m-1} \cos\left(\frac{\pi(2k+1)(m)}{N}\right)$$
- $\cos\left(\frac{\pi(2k+1)\frac{N}{2}}{N}\right) = \cos\left(\frac{\pi(2k+1)}{2}\right) = 0$
- $x_{2n-1}|_{n=0} = 0$

Calcul de la moitié des facteurs

$$\text{Rappel : } \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left(\frac{\pi(2k+1)(2n+1)}{2N}\right) =$$

$$\frac{1}{2 \cos\left(\frac{\pi(2k+1)}{2N}\right)} \sum_{n=0}^{\frac{N}{2}-1} (x_{2n+1} + x_{2n-1}) \cos\left(\frac{\pi(2k+1)n}{N}\right)$$

$$\blacksquare \forall k \in \llbracket 0; \frac{N}{2} - 1 \rrbracket, \cos\left(\frac{\pi(2(N-1-k)+1)}{2N}\right) = \cos\left(\frac{\pi(2N-2k-1)}{2N}\right) = \cos\left(\pi - \frac{\pi(2k+1)}{2N}\right) = -\cos\left(\frac{\pi(2k+1)}{2N}\right)$$

$$\blacksquare \forall k \in \llbracket 0; \frac{N}{2} - 1 \rrbracket, \cos\left(\frac{\pi(2(N-1-k)+1)n}{N}\right) = \cos\left(\frac{\pi(2N-2k-1)n}{N}\right) = \cos(2n\pi - \frac{\pi(2k+1)n}{N}) = \cos\left(\frac{\pi(2k+1)n}{N}\right)$$

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
1  ### CALCUL COEFFICIENTS ###
2
3  # Import des bibliothèques
4
5  import time
6  import numpy as np
7  import matplotlib.pyplot as plt
8
9  # Ouverture du document texte
10
11  f = open("aigle3.txt", 'r')
12
13  ### Programme Principal
14
15  # Convertir en liste
16
17  def OuvrEtConv (f) :
18
19      ...
20
21      entrée : str
22      sortie : list
23
24      but : prend en entrée la chaîne de caractères du type :
25      0.0066
26      0.0095
27      qui représente les ordonnées de la fonction tracée par le signal et qui renvoie la liste de ces valeurs
28
29      ...
30      c = f.read()
31      M = []
32      i = 0 #itération des caractères
33      z = 0 #compteur de valeurs
34
35      for k in range (len(c)) : # on compte ici le nombre de valeurs
36          if c[k] == '\n' :
37              z += 1
38
39      for j in range (z) : #j est donc l'indice de la valeur
40
41          P = 0 #valeur entière
42          D = 0 #valeur décimale
43          V = 0 #valeur complète
44          a = 0 #indique si la valeur est négative ou non
45
46          if c[i] == '-' : # valeur négative ou non
47              a += 1
48              i += 1
49
50          N = [] #liste comportant les caractères composant la partie entière
51          while c[i] != '.' : #on cherche la partie entière
52              N.append(c[i])
53              i += 1
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
52     i += 1
53     for k in range(len(N)) :
54         P += float(N[-k])*(10**(k+1))
55     i += 1
56     L = [] #même chose avec la partie décimale
57     while c[i] != '\n' :
58         L.append(c[i])
59     i += 1
60     for k in range(len(L)) :
61         D += int(L[k])*(10**(-k-1))
62     if a == 1 :
63         V = (-1)*(P + D)
64     else :
65         V = P + D
66     i += 1
67     if abs(V) >= 0.0001 : #pour prendre que les valeurs utiles
68         M.append(V)
69
70     return(M)
71
72 # Valeur moyenne
73
74 def VaMoy (L) :
75     ...
76
77     entrée : list
78     sortie : float
79
80
81     but : renvoie la valeur moyenne de la liste d'entrée
82
83     ...
84
85     a = 0 #somme des valeurs
86     for x in L:
87         a+=x
88
89     return a/len(L)
90
91 # Période
92
93 def Periode (L) :
94     ...
95
96
97     entrée : list
98     sortie : list
99
100
101     but : renvoie la liste des différentes périodes du signal (liste d'entrée)
102
103     ...
104
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
103
104 n = len(L)
105 f1 = 0 #début de la période
106 f2 = 0 #fin de la période
107 P = [] #liste des périodes
108 a = 0 #nombre de fois que la valeur moyenne est vue (pour une période)
109 VM = VaMoy(L)
110
111 for i in range(n-1) :
112     if (L[i] <= VM and L[i+1] > VM) or (L[i] >= VM and L[i+1] < VM) : #si la fonction passe par sa valeur moyenne
113         if a == 0 :
114             a += 1
115             f1 = 1
116         elif a == 1 :
117             a += 1
118         elif a == 2 :
119             f2 = i - 1
120             a = 0
121             P.append(f2-f1)
122
123 return P
124
125 # Échantillonnage
126
127 def Echan (L,points) :
128     ...
129
130     entrée : list
131     sortie : list
132
133     but : renvoie la liste échantillonnée, avec ici 'points' points par période, de la liste d'entrée
134     ...
135
136     # on essaie avec 'points' point par périodes
137
138     T = VaMoy(Periode(L))#fréquence moyenne
139     LE = [] #signal échantillonné
140     NbT = int(len(L)/T) #nombre de périodes
141     if NbT*points >= len(L) :
142         print("échantillonnage impossible, renvoi de la liste de départ non échantillonnée")
143     else :
144         for i in range (NbT*points) :
145             LE.append(L[int(i*(T//points))])
146
147     return LE
148
149 # Normaliser
150
151
152
153
154
155
```

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
154 # Normaliser
155
156 def SINor (L) :
157     ...
158
159     entrée : list
160     sortie : list
161
162     but : renvoie la liste d'entrée décalée verticalement, de telle sorte que sa valeur moyenne soit nulle
163
164     ...
165     LN = [] #nouvelle liste
166     VM = VaMoy(L)
167     for i in range (len(L)) :
168         LN.append(L[i] - VM)
169
170     return LN
171
172
173 # Pré-accentuer
174
175 def SiAcc (L,LN) : #on accentue certaines fréquences
176     ...
177
178     entrée : list,list
179     sortie : list
180
181     but : renvoie la liste d'entrée (LN) où les hautes fréquences sont accentuées
182
183     ...
184
185     K = 0.95 #facteur d'accentuation
186
187     LA = [VaMoy(L)] #on prend comme première valeur, la valeur moyenne du signal non normalisé
188     for i in range (len(L)-1) :
189         LA.append (LN[i+1]*K*LN[i])
190
191     return LA
192
193
194 # Trames
195
196 def Trame (L,LF) : #avec L la liste de base et LF la liste échantillonnée
197     ...
198
199     entrée : list,list
200     sortie : list(list)
201
202     but : renvoie la liste des différents échantillons de la liste d'entrée (LF)
203
204     ...
205
206
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

205     ...
206
207     T = VaMoy(Periode(L)) #période moyenne
208     Q = int(220.5/(len(LF)/len(L))) #longueur d'un trame
209     q = int(Q*2) #longueur de l'échantillon
210     end = int(len(LF)/Q) #nombre total possible de trames
211     LT = [] #Liste regroupant toutes les trames
212     K = 0
213
214     while K < end - 1 : #jusqu'a avoir fait le nombre maximal de trames
215         Ltemp = [] #trame
216         for i in range (0*K,Q*K + q) :
217             tra = L[i]*FoncFen(K-i,q)
218             'note : si problème par la suite, on peut essayer avec (K*Q+q) à la place de K'
219             Ltemp.append(tra)
220         LT.append(Ltemp)
221         K += 1
222
223     return LT
224
225 def FoncFen (n,K) :
226     ...
227
228     entrée : int,int
229     sortie : float
230
231     ...
232
233     return 0.54-0.46*np.cos(2*np.pi*((n-1)/(K-1)))
234
235 # Fourier
236
237 def Rajoute_0 (L) :
238     ...
239
240     ...
241
242     entrée : list
243     sortie : list
244
245     but : renvoie une liste dont le début est la liste d'entrée et la fin est des zéros, de telle sorte que la taille de la
246     liste soit une puissance de 2
247
248     ...
249
250     x = 0 #test
251     i = 0 #itération dans la liste
252     n = len(L)
253
254     while x == 0 : # on cherche la puissance de deux supérieure la plus proche
255         if n == 2**i :
256             x = 1
257             # et la puissance de deux supérieure la plus proche est 2 puissance i
258
259

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

255         x = 1
256         elif 2**i < n and 2**(i+1) > n : # si la puissance de deux supérieure la plus proche est 2 puissance i alors :
257             x = 1
258             i += 1
259         r = (2**i)-n #le nombre de 0 à rajouter
260         L_new = []
261         for k in L :
262             L_new.append(k)
263         for k in range(r) :
264             L_new.append(0)
265
266         return L_new
267
268     def FFT(x):
269         '''
270
271         entrée : list
272         sortie : list
273
274         but : renvoie la transformée de Fourier de x, version numpy
275
276         '''
277
278         # on se base sur la symétrie de la transformée de Fourier discrète, pour calculer la transformée de Fourier pour une liste
279         # de N valeurs, il suffit que de calculer la moitié des termes
280
281         N = len(x)
282
283         if N == 1: #on divise jusqu'à ce que les parties soient de taille 1
284
285             return x
286
287         else:
288             #la récursivité va ici permettre de diviser la somme en plusieurs parties puis de remonter l'algorithme
289             X_pair = FFT(x[::2])
290             X_impair = FFT(x[1::2])
291             facteur = np.exp(-2j*np.pi*np.arange(N)/ N)
292
293             X = np.concatenate([X_pair+facteur[int(N/2):]*X_impair,
294                                 X_pair+facteur[int(N/2):]*X_impair])
295
296             return X
297
298     def FFT2(x) :
299         '''
300
301         entrée : list
302         sortie : list
303
304         but : renvoie la transformée de Fourier de x, version numpy
305
306         '''

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
305 but : renvoie la transformée de Fourier de x, version lisible
306 ...
307
308
309 N = len(x)
310
311 if N == 1 : #on divise la liste jusqu'à avoir une liste de taille 1
312     return x
313
314 else :
315     X_pair = FFT2(x[::2])
316     X_impair = FFT2(x[1::2])
317
318     facteurs = [] #on crée nos facteurs
319     for i in range (N) :
320         facteurs.append(np.exp(-2j*np.pi*i/N))
321
322     X1 = []
323     X2 = []
324     for i in range (len(X_pair)) :
325         X1.append(X_pair[i]+facteurs[:,int(N/2)][i]*X_impair[i])
326         X2.append(X_pair[i]+facteurs[int(N/2):][i]*X_impair[i])
327
328     return X1 + X2
329
330 def ToutesTFD (L) :
331     ...
332
333     entrée : list(list)
334     sortie : list
335
336     but : renvoie la transformée de Fourier discrète appliquée à chaque liste dans la liste d'entrée
337     ...
338
339     TFDS = []
340     for x in L :
341         tf = FFT2(Rajoute_0(x)) #transformée de Fourier appliquée à une liste
342         TFDS.append(tf)
343
344     return TFDS
345
346 def TFD(x) :
347     ...
348
349     entrée : list
350     sortie : list
351
352     but : renvoie la transformée de Fourier de x (version lente)
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
356 but : renvoie la transformée de Fourier de x (version lente)
357
358 ...
359
360 N = len(x)
361
362 Coefs = []
363 for r in range(N) :
364     Co = 0
365     for k in range(N) :
366         Co += x[k]*np.exp(-2j*np.pi*r*k/N)
367     Coefs.append(Co)
368
369 return Coefs
370
371 def ToutesTFDLentes (L) :
372
373     ...
374
375     entrée : list(list)
376     sortie : list
377
378     but : renvoie la transformée de Fourier discrète appliquée à chaque liste dans la liste d'entrée
379
380     ...
381
382     TFDS = []
383     for x in L :
384         tf = TFD(x) #transformée de Fourier appliquée à une liste
385         TFDS.append(tf)
386
387     return TFDS
388
389 # Périodogramme
390
391 def Per(L) :
392
393     ...
394
395     entrée : list(list)
396     sortie : list(list)
397
398     but : renvoie le périodogramme de chaque sous-liste de la liste d'entrée
399
400     ...
401
402     # On calcule le périodogramme pour toutes les sous listes, c'est-à-dire qu'on calcule le module de chaque valeur au carré,
403     # et on divise cette valeur par la taille de la sous-liste associée
404
405     Pe = [] #liste des périodogrammes
406
407     for x in L :
```


Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

406     for x in L :
407         PP = []
408         for i in range (len(x)) :
409             p = (1/len(x))*((np.abs(x[i]))**2)
410             PP.append(p)
411         Pe.append(PP)
412
413     return Pe
414
415 # Mel filterbank
416
417 def Mel(f) :
418     '''
419
420     entrée : float
421     sortie : float
422
423     but : renvoie dans le domaine de fréquences Mel la fréquence f
424
425     '''
426
427     return 1125*np.log(1+(f/700))
428
429 def Melinv(m) :
430     '''
431
432     entrée : float
433     sortie : float
434
435     but : renvoie dans le domaine fréquentiel classique la fréquence f (Mel)
436
437     '''
438
439     return 700*(np.exp(m/1125)-1)
440
441 def Melfilterbank(L,n,sr) :
442     '''
443
444     entrée : list,int,float
445     sortie : list(list)
446
447     but : renvoie la liste des filtres du melfilterbank
448
449     '''
450
451     #n le nombre de filtres
452
453     Fmin = 0
454     Fmax = len(L)
455     F = np.linspace(Fmin, Fmax, n+1)
456     F = F[1:]
457     F = F*sr/1000
458     F = F*2
459     F = F/2
460     F = F*1000/sr
461     F = F/2
462     F = F*1000/sr
463     F = F/2
464     F = F*1000/sr
465     F = F/2
466     F = F*1000/sr
467     F = F/2
468     F = F*1000/sr
469     F = F/2
470     F = F*1000/sr
471     F = F/2
472     F = F*1000/sr
473     F = F/2
474     F = F*1000/sr
475     F = F/2
476     F = F*1000/sr
477     F = F/2
478     F = F*1000/sr
479     F = F/2
480     F = F*1000/sr
481     F = F/2
482     F = F*1000/sr
483     F = F/2
484     F = F*1000/sr
485     F = F/2
486     F = F*1000/sr
487     F = F/2
488     F = F*1000/sr
489     F = F/2
490     F = F*1000/sr
491     F = F/2
492     F = F*1000/sr
493     F = F/2
494     F = F*1000/sr
495     F = F/2
496     F = F*1000/sr
497     F = F/2
498     F = F*1000/sr
499     F = F/2
500     F = F*1000/sr
501     F = F/2
502     F = F*1000/sr
503     F = F/2
504     F = F*1000/sr
505     F = F/2
506     F = F*1000/sr
507     F = F/2
508     F = F*1000/sr
509     F = F/2
510     F = F*1000/sr
511     F = F/2
512     F = F*1000/sr
513     F = F/2
514     F = F*1000/sr
515     F = F/2
516     F = F*1000/sr
517     F = F/2
518     F = F*1000/sr
519     F = F/2
520     F = F*1000/sr
521     F = F/2
522     F = F*1000/sr
523     F = F/2
524     F = F*1000/sr
525     F = F/2
526     F = F*1000/sr
527     F = F/2
528     F = F*1000/sr
529     F = F/2
530     F = F*1000/sr
531     F = F/2
532     F = F*1000/sr
533     F = F/2
534     F = F*1000/sr
535     F = F/2
536     F = F*1000/sr
537     F = F/2
538     F = F*1000/sr
539     F = F/2
540     F = F*1000/sr
541     F = F/2
542     F = F*1000/sr
543     F = F/2
544     F = F*1000/sr
545     F = F/2
546     F = F*1000/sr
547     F = F/2
548     F = F*1000/sr
549     F = F/2
550     F = F*1000/sr
551     F = F/2
552     F = F*1000/sr
553     F = F/2
554     F = F*1000/sr
555     F = F/2
556     F = F*1000/sr
557     F = F/2
558     F = F*1000/sr
559     F = F/2
560     F = F*1000/sr
561     F = F/2
562     F = F*1000/sr
563     F = F/2
564     F = F*1000/sr
565     F = F/2
566     F = F*1000/sr
567     F = F/2
568     F = F*1000/sr
569     F = F/2
570     F = F*1000/sr
571     F = F/2
572     F = F*1000/sr
573     F = F/2
574     F = F*1000/sr
575     F = F/2
576     F = F*1000/sr
577     F = F/2
578     F = F*1000/sr
579     F = F/2
580     F = F*1000/sr
581     F = F/2
582     F = F*1000/sr
583     F = F/2
584     F = F*1000/sr
585     F = F/2
586     F = F*1000/sr
587     F = F/2
588     F = F*1000/sr
589     F = F/2
590     F = F*1000/sr
591     F = F/2
592     F = F*1000/sr
593     F = F/2
594     F = F*1000/sr
595     F = F/2
596     F = F*1000/sr
597     F = F/2
598     F = F*1000/sr
599     F = F/2
600     F = F*1000/sr
601     F = F/2
602     F = F*1000/sr
603     F = F/2
604     F = F*1000/sr
605     F = F/2
606     F = F*1000/sr
607     F = F/2
608     F = F*1000/sr
609     F = F/2
610     F = F*1000/sr
611     F = F/2
612     F = F*1000/sr
613     F = F/2
614     F = F*1000/sr
615     F = F/2
616     F = F*1000/sr
617     F = F/2
618     F = F*1000/sr
619     F = F/2
620     F = F*1000/sr
621     F = F/2
622     F = F*1000/sr
623     F = F/2
624     F = F*1000/sr
625     F = F/2
626     F = F*1000/sr
627     F = F/2
628     F = F*1000/sr
629     F = F/2
630     F = F*1000/sr
631     F = F/2
632     F = F*1000/sr
633     F = F/2
634     F = F*1000/sr
635     F = F/2
636     F = F*1000/sr
637     F = F/2
638     F = F*1000/sr
639     F = F/2
640     F = F*1000/sr
641     F = F/2
642     F = F*1000/sr
643     F = F/2
644     F = F*1000/sr
645     F = F/2
646     F = F*1000/sr
647     F = F/2
648     F = F*1000/sr
649     F = F/2
650     F = F*1000/sr
651     F = F/2
652     F = F*1000/sr
653     F = F/2
654     F = F*1000/sr
655     F = F/2
656     F = F*1000/sr
657     F = F/2
658     F = F*1000/sr
659     F = F/2
660     F = F*1000/sr
661     F = F/2
662     F = F*1000/sr
663     F = F/2
664     F = F*1000/sr
665     F = F/2
666     F = F*1000/sr
667     F = F/2
668     F = F*1000/sr
669     F = F/2
670     F = F*1000/sr
671     F = F/2
672     F = F*1000/sr
673     F = F/2
674     F = F*1000/sr
675     F = F/2
676     F = F*1000/sr
677     F = F/2
678     F = F*1000/sr
679     F = F/2
680     F = F*1000/sr
681     F = F/2
682     F = F*1000/sr
683     F = F/2
684     F = F*1000/sr
685     F = F/2
686     F = F*1000/sr
687     F = F/2
688     F = F*1000/sr
689     F = F/2
690     F = F*1000/sr
691     F = F/2
692     F = F*1000/sr
693     F = F/2
694     F = F*1000/sr
695     F = F/2
696     F = F*1000/sr
697     F = F/2
698     F = F*1000/sr
699     F = F/2
700     F = F*1000/sr
701     F = F/2
702     F = F*1000/sr
703     F = F/2
704     F = F*1000/sr
705     F = F/2
706     F = F*1000/sr
707     F = F/2
708     F = F*1000/sr
709     F = F/2
710     F = F*1000/sr
711     F = F/2
712     F = F*1000/sr
713     F = F/2
714     F = F*1000/sr
715     F = F/2
716     F = F*1000/sr
717     F = F/2
718     F = F*1000/sr
719     F = F/2
720     F = F*1000/sr
721     F = F/2
722     F = F*1000/sr
723     F = F/2
724     F = F*1000/sr
725     F = F/2
726     F = F*1000/sr
727     F = F/2
728     F = F*1000/sr
729     F = F/2
730     F = F*1000/sr
731     F = F/2
732     F = F*1000/sr
733     F = F/2
734     F = F*1000/sr
735     F = F/2
736     F = F*1000/sr
737     F = F/2
738     F = F*1000/sr
739     F = F/2
740     F = F*1000/sr
741     F = F/2
742     F = F*1000/sr
743     F = F/2
744     F = F*1000/sr
745     F = F/2
746     F = F*1000/sr
747     F = F/2
748     F = F*1000/sr
749     F = F/2
750     F = F*1000/sr
751     F = F/2
752     F = F*1000/sr
753     F = F/2
754     F = F*1000/sr
755     F = F/2
756     F = F*1000/sr
757     F = F/2
758     F = F*1000/sr
759     F = F/2
760     F = F*1000/sr
761     F = F/2
762     F = F*1000/sr
763     F = F/2
764     F = F*1000/sr
765     F = F/2
766     F = F*1000/sr
767     F = F/2
768     F = F*1000/sr
769     F = F/2
770     F = F*1000/sr
771     F = F/2
772     F = F*1000/sr
773     F = F/2
774     F = F*1000/sr
775     F = F/2
776     F = F*1000/sr
777     F = F/2
778     F = F*1000/sr
779     F = F/2
780     F = F*1000/sr
781     F = F/2
782     F = F*1000/sr
783     F = F/2
784     F = F*1000/sr
785     F = F/2
786     F = F*1000/sr
787     F = F/2
788     F = F*1000/sr
789     F = F/2
790     F = F*1000/sr
791     F = F/2
792     F = F*1000/sr
793     F = F/2
794     F = F*1000/sr
795     F = F/2
796     F = F*1000/sr
797     F = F/2
798     F = F*1000/sr
799     F = F/2
800     F = F*1000/sr
801     F = F/2
802     F = F*1000/sr
803     F = F/2
804     F = F*1000/sr
805     F = F/2
806     F = F*1000/sr
807     F = F/2
808     F = F*1000/sr
809     F = F/2
810     F = F*1000/sr
811     F = F/2
812     F = F*1000/sr
813     F = F/2
814     F = F*1000/sr
815     F = F/2
816     F = F*1000/sr
817     F = F/2
818     F = F*1000/sr
819     F = F/2
820     F = F*1000/sr
821     F = F/2
822     F = F*1000/sr
823     F = F/2
824     F = F*1000/sr
825     F = F/2
826     F = F*1000/sr
827     F = F/2
828     F = F*1000/sr
829     F = F/2
830     F = F*1000/sr
831     F = F/2
832     F = F*1000/sr
833     F = F/2
834     F = F*1000/sr
835     F = F/2
836     F = F*1000/sr
837     F = F/2
838     F = F*1000/sr
839     F = F/2
840     F = F*1000/sr
841     F = F/2
842     F = F*1000/sr
843     F = F/2
844     F = F*1000/sr
845     F = F/2
846     F = F*1000/sr
847     F = F/2
848     F = F*1000/sr
849     F = F/2
850     F = F*1000/sr
851     F = F/2
852     F = F*1000/sr
853     F = F/2
854     F = F*1000/sr
855     F = F/2
856     F = F*1000/sr
857     F = F/2
858     F = F*1000/sr
859     F = F/2
860     F = F*1000/sr
861     F = F/2
862     F = F*1000/sr
863     F = F/2
864     F = F*1000/sr
865     F = F/2
866     F = F*1000/sr
867     F = F/2
868     F = F*1000/sr
869     F = F/2
870     F = F*1000/sr
871     F = F/2
872     F = F*1000/sr
873     F = F/2
874     F = F*1000/sr
875     F = F/2
876     F = F*1000/sr
877     F = F/2
878     F = F*1000/sr
879     F = F/2
880     F = F*1000/sr
881     F = F/2
882     F = F*1000/sr
883     F = F/2
884     F = F*1000/sr
885     F = F/2
886     F = F*1000/sr
887     F = F/2
888     F = F*1000/sr
889     F = F/2
890     F = F*1000/sr
891     F = F/2
892     F = F*1000/sr
893     F = F/2
894     F = F*1000/sr
895     F = F/2
896     F = F*1000/sr
897     F = F/2
898     F = F*1000/sr
899     F = F/2
900     F = F*1000/sr
901     F = F/2
902     F = F*1000/sr
903     F = F/2
904     F = F*1000/sr
905     F = F/2
906     F = F*1000/sr
907     F = F/2
908     F = F*1000/sr
909     F = F/2
910     F = F*1000/sr
911     F = F/2
912     F = F*1000/sr
913     F = F/2
914     F = F*1000/sr
915     F = F/2
916     F = F*1000/sr
917     F = F/2
918     F = F*1000/sr
919     F = F/2
920     F = F*1000/sr
921     F = F/2
922     F = F*1000/sr
923     F = F/2
924     F = F*1000/sr
925     F = F/2
926     F = F*1000/sr
927     F = F/2
928     F = F*1000/sr
929     F = F/2
930     F = F*1000/sr
931     F = F/2
932     F = F*1000/sr
933     F = F/2
934     F = F*1000/sr
935     F = F/2
936     F = F*1000/sr
937     F = F/2
938     F = F*1000/sr
939     F = F/2
940     F = F*1000/sr
941     F = F/2
942     F = F*1000/sr
943     F = F/2
944     F = F*1000/sr
945     F = F/2
946     F = F*1000/sr
947     F = F/2
948     F = F*1000/sr
949     F = F/2
950     F = F*1000/sr
951     F = F/2
952     F = F*1000/sr
953     F = F/2
954     F = F*1000/sr
955     F = F/2
956     F = F*1000/sr
957     F = F/2
958     F = F*1000/sr
959     F = F/2
960     F = F*1000/sr
961     F = F/2
962     F = F*1000/sr
963     F = F/2
964     F = F*1000/sr
965     F = F/2
966     F = F*1000/sr
967     F = F/2
968     F = F*1000/sr
969     F = F/2
970     F = F*1000/sr
971     F = F/2
972     F = F*1000/sr
973     F = F/2
974     F = F*1000/sr
975     F = F/2
976     F = F*1000/sr
977     F = F/2
978     F = F*1000/sr
979     F = F/2
980     F = F*1000/sr
981     F = F/2
982     F = F*1000/sr
983     F = F/2
984     F = F*1000/sr
985     F = F/2
986     F = F*1000/sr
987     F = F/2
988     F = F*1000/sr
989     F = F/2
990     F = F*1000/sr
991     F = F/2
992     F = F*1000/sr
993     F = F/2
994     F = F*1000/sr
995     F = F/2
996     F = F*1000/sr
997     F = F/2
998     F = F*1000/sr
999     F = F/2
1000    F = F*1000/sr

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

457 Fmax = len(L)
458 Mmin = Mel(Fmin)
459 Mmax = Mel(Fmax)
460
461 M = [] #Liste de points entre Mmin et Mmax
462 d = Mmax - Mmin
463 for i in range (n+2) :
464     M.append(Mmin + (i)*d/(n+1))
465
466 H = [] #même chose convertie en Hertz
467 for x in M :
468     H.append(Melinv(x))
469
470 filterbank = []
471 for m in range (n) :
472     Temporary = [] #liste temporaire des points d'un filtre
473     for k in range (int(H[-1])) : #création des filtres triangulaires
474         if k < H[m] :
475             Temporary.append(0)
476         elif k >= H[m] and k < H[m+1] :
477             Temporary.append((k-H[m])/(H[m+1]-H[m]))
478         elif k == H[m+1] :
479             Temporary.append(1)
480         elif k > H[m+1] and k <= H[m+2] :
481             Temporary.append((H[m+2]-k)/(H[m+2]-H[m+1]))
482         elif k > H[m+2] :
483             Temporary.append(0)
484     filterbank.append(Temporary)
485
486 return filterbank
487
488 def ValeursMaxListes(L) :
489     ...
490
491     entrée : list(list)
492     sortie : list
493
494     but : renvoie la liste constituée des valeurs maximales entre chaque liste
495
496     ...
497
498     #on vérifie que les listes de sont de la même taille
499     liste tailles = []
500     for i in range (len(L)) :
501         liste tailles.append(len(L[i]))
502     for x in liste tailles :
503         if liste tailles[0] != x : #dès qu'une taille de liste est différente de la première, on renvoie rien, en disant que
504         les listes ne sont pas de la même taille
505         print("les listes ne sont pas de la même taille")
506         return None
507
508     liste valeursmax = []

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
507
508 liste_valeursmax = []
509 for k in range (len(L[0])) :
510     ValeursIndexK = [] #on créé la liste des valeurs d'indice k
511     for i in range (len(L)) :
512         ValeursIndexK.append(L[i][k])
513     liste_valeursmax.append(max(ValeursIndexK)) #on ajoute à la liste finale le maximum des valeurs d'indice k
514
515 return liste_valeursmax
516
517 def Melfiltrage(L,n,sr) :
518     ...
519
520     entrée : list(list),int,float
521     sortie : list(list)
522
523     but : renvoie la liste des sous-listes de la liste d'entrée filtrées par le melfilterbank
524
525     ...
526
527     #on crée nos filtres
528
529     Filtres = []
530     for i in range (len(L)) :
531         Filtres.append(Melfilterbank(L[i],n,sr))
532
533     #on prend les valeurs maximales des filtres pour chaque trame, pour pouvoir filtrer chaque trame
534
535     Filtres_max = []
536     for i in range (len(Filtres)) :
537         Filtres_max.append(ValeursMaxListes(Filtres[i]))
538
539     #on filtre, c'est-à-dire qu'on multiplie chaque fréquence de chaque trame par la valeur correspondante du filtre associé
540     (valeur entre 0 et 1)
541
542     Listes_filtered = []
543     for i in range (len(L)) :
544         Liste_filtered_i = []
545         for j in range (len(L[i])-1) :
546             Liste_filtered_i.append(L[i][j]*Filtres_max[i][j])
547         Listes_filtered.append(Liste_filtered_i)
548
549     return Listes_filtered
550
551 # Logarithme
552
553 def Amplification(L) :
554     ...
555
556     entrée : list(list),int
557     sortie : list(list)
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
557 entrée : list(list),int
558 sortie : list(list)
559
560 but : renvoie la liste des sous-listes de la liste d'entrée amplifiées
561 ...
562
563
564 #le signal ayant des valeurs trop petites, le logarithme renvoie une valeur négative, on amplifie donc le signal
565
566 Liste_min = []
567 for x in L :
568     Liste_min.append(min(x))
569
570 if min(Liste_min) >= 1 :
571     return L
572
573
574 else :
575     Amp = 1 - min(Liste_min)
576
577     L_ampli = [] #on les amplifie
578     for i in range (len(L)) :
579         L_ampli_i = []
580         for j in range (len(L[i])) :
581             L_ampli_i.append(Amp + L[i][j])
582         L_ampli.append(L_ampli_i)
583
584     return L_ampli
585
586 def Passagelog (L) :
587     ...
588
589
590 entrée : list(list)
591 sortie : list(list)
592
593 but : renvoie la liste des sous-listes de la liste d'entrée passées au logarithme
594 ...
595
596
597 #on passe au logarithme une liste de liste
598
599 L_log = []
600 for i in range (len(L)) :
601     L_temp_log = []
602     for j in range (len(L[i])) :
603         if L[i][j] != 0 :
604             L_temp_log.append(np.log(L[i][j]))
605     L_log.append(L_temp_log)
606
607 return L_log
608
609 # Transformée en cosinus discrète
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
608 # Transformée en cosinus discrète
609
610 def DCT(x) :
611     ...
612
613     entrée : list
614     sortie : list
615
616     but : renvoie la transformée en cosinus discrète de x (version lente)
617
618     ...
619
620     N = len(x) #taille de la liste
621
622     Coefs = []
623     for r in range (N) :
624         Co = 0
625         for k in range (N) :
626             Co += x[k]*np.cos(np.pi*r*(k+1/2)/N)
627         Coefs.append(Co)
628
629     return Coefs
630
631 def FDCT(x) :
632     ...
633
634     entrée : list
635     sortie : list
636
637     but : renvoie la transformée en cosinus discrète de x, version rapide
638
639     ...
640
641     N = len(x) #taille
642
643     if N == 1 :
644         return x
645
646     else :
647
648         #on divise la liste
649         X1 = []
650         X2 = []
651         for i in range (int(N/2)) :
652
653             X1.append(x[i]+x[-i-1])
654             X2.append((x[i]-x[-i-1])/(np.cos((i+0.5)*np.pi/N)*2))
655
656         X1 = FDCT(X1)
657         X2 = FDCT(X2)
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
659     X1 = FDCT(X1)
660     X2 = FDCT(X2)
661
662     X = []
663     for i in range (int(N/2)-1) :
664         X.append(X1[i])
665         X.append(X2[i] + X2[i+1])
666     X.append(X1[-1])
667     X.append(X2[-1])
668
669     return X
670
671
672 def ToutesDCT(L) :
673     '''
674
675     entrée : list(list)
676     sortie : list(list)
677
678     but : renvoie la liste des transformées en cosinus discrètes des sous-listes de la liste d'entrée (rapide)
679
680     '''
681
682     L_nouveau = []
683
684     for x in L :
685         dct = FDCT(Rajoute_0(x))
686         L_nouveau.append(dct)
687
688     return L_nouveau
689
690
691 def ToutesDCTLentes(L) :
692     '''
693
694     entrée : list(list)
695     sortie : list(list)
696
697     but : renvoie la liste des transformées en cosinus discrètes des sous-listes de la liste d'entrée (lente)
698
699     '''
700
701     L_nouveau = []
702
703     for x in L :
704         dct = DCT(Rajoute_0(x))
705         L_nouveau.append(dct)
706
707     return L_nouveau
708
709
710 # Coefficients
711
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

710 # Coefficients
711
712 def MFCC(L,c) :
713     ...
714
715     entrée : list(list)
716     sortie : list(list)
717
718     but : renvoie et affiche les c premiers mfcc en fonction du temps
719
720     ...
721
722     Liste_coefs = [] #on crée la liste des coefficients en fonction du temps
723     for i in range (c) :
724         Liste_coef_c = []
725         for j in range (len(L)) :
726             Liste_coef_c.append(L[j][i])
727         Liste_coefs.append(Liste_coef_c)
728
729     plt.imshow(Liste_coefs,origin = 'lower',cmap='jet',aspect="auto")
730     plt.colorbar(label="Valeur coefficients")
731     plt.gcf().set_size_inches(plt.gcf().get_size_inches()*1.5)
732     plt.xlabel("Fenêtre", fontsize='xx-large')
733     plt.ylabel("Numéro Coefficient", fontsize='xx-large')
734     plt.show()
735
736     return Liste_coefs
737
738
739 # Exécution
740
741 def Exe(f,c,e) :
742     ...
743
744     entrée : str,int,int
745     sortie : list(list)
746
747     but : execution du programme, f étant l'entrée, c le nombre de coefficients et e le nombre de points par période pour
748     l'échantillonnage
749
750     ...
751
752     L = OuvrEtConv(f)
753     if L != [] :
754         LE = Echan(L,e)
755         print("signal échantillonné à ", len(LE),"éléments au lieu de ", len(L))
756         LN = SiNor(LE)
757         print("signal normalisé")
758         LA = SiAcc(LE,LN)
759         print("signal pré accentué")
760         LT = Trame(L,LA)
761         print("signal traité")

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
760     LT = Trame(L,LA)
761     print("signal séparé")
762     TFDS = ToutesTFD(LT)
763     print("transformés calculés")
764     Pe = Per(TFDS)
765     print("périodogramme calculé")
766     M = Malfiltrage(Pe,10,44150)
767     print("fréquences filtrées")
768     Log = PassageLog(Amplification(M))
769     print("signal passé au logarithme")
770     dct = ToutesDCT(Log)
771     print("transformé en cos")
772     mfcc = MFCC(dct,c)
773     print("coefficients calculés")
774
775     return mfcc
776
777     ### Programme Annexe
778
779     # Test log
780
781     def test_log(L) :
782         ...
783
784         entrée : list(list)
785         sortie : int,int
786
787         but : renvoie le nombre de valeurs interdites pour le logarithme
788
789         ...
790
791         zeros = 0
792         negatifs = 0
793
794         for sous_liste in L :
795             for x in sous_liste :
796                 if x == 0 :
797                     zeros += 1
798                 if x < 0 :
799                     negatifs += 1
800
801         return zeros,negatifs
802
803     # Temps comparatif
804
805     def ListeMoins (L,d) :
806         ...
807
808         entrée : list,int
809         sortie : list
810
811         ...
```


Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
811     sortie : list
812
813     but : renvoie la liste L avec les d derniers termes enlevés
814     ...
815
816     if d < len(L) :
817         return L[0:len(L)-d]
818     else :
819         return L
820
821
822 def ExeC (E,L) :
823     ...
824
825     entrée : bool,list
826
827     but : execution du programme avec ou sans fft
828     ...
829
830
831     if E == True : #execution avec fft
832         if L!= [] :
833             LN = SiNor(L)
834             LA = SiAcc(L,LN)
835             LT = Trame(L,LA)
836             TFDS = ToutesTFD(LT)
837             Pe = Per(TFDS)
838             M = Melfiltrage(Pe,10,44150)
839             Log = Passagelog(M)
840             dct = ToutesDCT(Log)
841
842
843
844     else : #execution sans fft
845         if L!= [] :
846             LN = SiNor(L)
847             LA = SiAcc(L,LN)
848             LT = Trame(L,LA)
849             TFDS = ToutesTFDlentes(LT)
850             Pe = Per(TFDS)
851             M = Melfiltrage(Pe,10,44150)
852             Log = Passagelog(M)
853             dct = ToutesDCT(Log)
854
855
856 def ExeE (E,L) :
857     ...
858
859     entrée : bool,list
860
861     but : execution du programme avec ou sans échantillonnage
862     ...
863
```

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914

'''
if E == True : #exécution avec échantillonnage
    if L!= [] :
        LE = Echan(L,4)
        LN = SiNor(LE)
        LA = SiAcc(LE,LN)
        LT = Trame(L,LA)
        TFDS = ToutesTFD(LT)
        Pe = Per(TFDS)
        M = Melfiltrage(Pe,10,44150)
        Log = PassageLog(M)
        dct = ToutesDCT(Log)

    else : #exécution sans échantillonnage
        if L!= [] :
            LN = SiNor(L)
            LA = SiAcc(L,LN)
            LT = Trame(L,LA)
            TFDS = ToutesTFD(LT)
            Pe = Per(TFDS)
            M = Melfiltrage(Pe,10,44150)
            Log = PassageLog(M)
            dct = ToutesDCT(Log)

def ExeTfft(E,L) :
    '''
    entrée : bool,list
    but : exécution de la tfd ou de la fft
    '''

    if E == False : #tfd
        if L!=[] :
            LN = SiNor(L)
            LA = SiAcc(L,LN)
            fft = TFD(LA)
        else : #fft
            if L!=[] :
                LN = SiNor(L)
                LA = SiAcc(L,LN)
                fft = FFT2(Rajoute_0(LA))

def Comparaisonfft_sanstrames (f,n) :
    '''
    -----

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965

entrée : str,int

but : créer un graphe comparatif des méthodes fft et tfd
...

LtempsE = [] #liste des temps avec fft
LtempsS = [] #liste des temps sans fft
Ltaille = [] #liste des tailles de liste
L = OuvrEtConv(f)

for i in range(n) :
    d = int(len(L)/n)*i #distance à retirer chaque fois
    LMod = ListeMoins(L,d) #liste avec cette distance en moins

    Ltaille.append(len(LMod)) #on ajoute à la liste des abscisses la taille de la liste

    s1 = time.time() #temps d'exécution avec échantillon
    ExeTfft(True,LMod)
    LtempsE.append(time.time() - s1)
    print("Bon pour E", i+1, "fois")

    s2 = time.time() #temps d'exécution sans échantillon
    ExeTfft(False,LMod)
    LtempsS.append(time.time() - s2)
    print("Bon pour S", i+1, "fois")

plt.plot(Ltaille,LtempsE)
plt.plot(Ltaille,LtempsS)
plt.legend(['Transformée de Fourier rapide','Décomposition en série de Fourier finie'],fontsize='xx-large')
plt.title("Temps comparatif de calcul de la transformée de Fourier rapide et discrète",fontsize='xx-large')
plt.xlabel("Taille de liste (n)",fontsize='xx-large')
plt.ylabel("Temps d'exécution (s)",fontsize='xx-large')
plt.show()

def Comparaisonfft (f,n) :
    ...

    entrée : str,int

    but : créer un graphe comparatif des méthodes fft et tfd
    ...

    LtempsE = [] #liste des temps avec fft
    LtempsS = [] #liste des temps sans fft
    Ltaille = [] #liste des tailles de liste
    L = OuvrEtConv(f)

    for i in range(n) :
```

Reconnaissance
de chant
d'oiseau

Présentation
du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

964 for i in range(n):
965     d = int(len(L)/n)*i #distance à retirer chaque fois
966     LMod = ListeMoins(L,d) #liste avec cette distance en moins
967
968     Ltaille.append(len(LMod)) #on ajoute à la liste des abscisses la taille de la liste
969
970     s1 = time.time() #temps d'exécution avec échantillonnage
971     ExeC(True,LMod)
972     LtempsE.append(time.time() - s1)
973     print("Bon pour E", i+1, "fois")
974
975     s2 = time.time() #temps d'exécution sans échantillonnage
976     ExeC(False,LMod)
977     LtempsS.append(time.time() - s2)
978     print("Bon pour S", i+1, "fois")
979
980 plt.plot(Ltaille,LtempsE)
981 plt.plot(Ltaille,LtempsS)
982 plt.legend(['Transformée de Fourier rapide','Décomposition en série de Fourier finie'],fontsize='xx-large')
983 plt.title("Temps comparatif de calcul avec et sans transformée rapide",fontsize='xx-large')
984 plt.xlabel("Taille de liste (n)",fontsize='xx-large')
985 plt.ylabel("Temps d'exécution (s)",fontsize='xx-large')
986 plt.show()
987
988 def ComparaisonE(f,n):
989     '''
990
991     entrée : str,int
992
993     but : créer un graphe comparatif des méthodes avec échantillonnage et sans
994
995     '''
996
997     LtempsE = [] #liste des temps avec échantillonnage
998     LtempsS = [] #liste des temps sans échantillonnage
999     Ltaille = [] #liste des tailles de liste
1000     L = OuvrEtConv(f)
1001
1002
1003     for i in range(n):
1004         d = int(len(L)/n)*i #distance à retirer chaque fois
1005         LMod = ListeMoins(L,d) #liste avec cette distance en moins
1006
1007         Ltaille.append(len(LMod)) #on ajoute à la liste des abscisses la taille de la liste
1008
1009         s1 = time.time() #temps d'exécution avec échantillonnage
1010         ExeC(True,LMod)
1011         LtempsE.append(time.time() - s1)
1012         print("Bon pour E", i+1, "fois")
1013
1014         s2 = time.time() #temps d'exécution sans échantillonnage
1015         ExeC(False,LMod)
1016         LtempsS.append(time.time() - s2)
1017         print("Bon pour S", i+1, "fois")

```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et pré-réglages

Séparation

Fourier

Filtrage

Logarithme et cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
1015     ExeE(False,LMod)
1016     LtempsS.append(time.time() - s2)
1017     print("Bon pour S", i+1, "fois")
1018
1019     plt.plot(Ltaille,LtempsE)
1020     plt.plot(Ltaille,LtempsS)
1021     plt.legend(['échantillonné','non échantillonné'],fontsize='xx-large')
1022     plt.title("Temps comparatif de calcul avec et sans échantillonnage",fontsize='xx-large')
1023     plt.xlabel("Taille de liste (n)",fontsize='xx-large')
1024     plt.ylabel("Temps d'exécution (s)",fontsize='xx-large')
1025     plt.show()
1026
1027 # Extraction
1028
1029 def Extrac(L) :
1030     '''
1031
1032     entrée : list(list)
1033
1034     but : créer un fichier texte avec les coefficients de fréquence mel
1035
1036     '''
1037
1038     Fichier = open("Pigeon3.txt",'a')
1039     for x in L :
1040         for k in x :
1041             Fichier.write(str(k))
1042             Fichier.write('\n')
1043             Fichier.write('\n')
1044             Fichier.write('\n')
1045     Fichier.close()
1046
1047 # Complexité
1048
1049 def N_carre(n) :
1050     '''
1051
1052     entrée : int
1053
1054     but : simule une fonction de complexité n carré, analogue à la décomposition en série de Fourier finie
1055
1056     '''
1057
1058     Simu = []
1059     for i in range (n) :
1060         Valeur = 0
1061         for k in range (n) :
1062             Valeur += k*np.exp(-2j*np.pi*k*i)
1063         Simu.append(Valeur)
1064
1065 def N_logN(n) :
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```
1066 def N_logN(n) :
1067     '''
1068
1069     entrée : int
1070
1071     but : simule une fonction de complexité n*log(n), analogue à la transformée de Fourier rapide
1072     '''
1073
1074     Simu = []
1075     for i in range (n) :
1076         Valeur = 0
1077         for k in range (int(np.log(n))) :
1078             Valeur += k*np.exp(-2j*np.pi*k*i)
1079         Simu.append(Valeur)
1080
1081
1082 def Liste_entiers(n) :
1083     '''
1084
1085     entrée : int
1086     sortie : list
1087
1088     but : renvoie la liste des entiers de 1 à n
1089     '''
1090
1091     Liste = []
1092     for i in range (n) :
1093         Liste.append(i+1)
1094
1095     return Liste
1096
1097 def ComparaisonComp(N,n) :
1098     '''
1099
1100     entrée : int,int
1101
1102     but : affiche un graphique comparatif entre les temps d'exécution de fonctions ayant pour complexité N² et Nlog(N)
1103     '''
1104
1105     L = Liste_entiers(N)
1106     Ltaille = []
1107     LtempsC = []
1108     LtempsL = []
1109
1110     for i in range (n) :
1111         d = int((len(L)/n)*i) #distance à retirer chaque fois
1112         LMod = ListeMoins(L,d) #liste avec cette distance en moins
1113
1114
```

Reconnaissance de chant d'oiseau

Présentation du sujet

Principe

Etude

Plan du calcul

Acquisition et
pré-réglages

Séparation

Fourier

Filtrage

Logarithme et
cosinus

Application

Problème 1

Problème 2

Filtrage

Problème 3

Coefficients

Conclusion

Annexe

Démonstration

Code

```

1112 LtempsC = []
1113 LtempsL = []
1114
1115 for i in range(n):
1116     d = int((len(L)/n)*i) #distance à retirer chaque fois
1117     LMod = ListeMoins(L,d) #liste avec cette distance en moins
1118
1119     Ltaille.append(len(LMod)) #on ajoute à la liste des abscisses la taille de la liste
1120
1121     s1 = time.time() #temps d'exécution de complexité en carré
1122     N_carré(len(LMod))
1123     LtempsC.append(time.time() - s1)
1124     print("Bon pour C", i+1, "fois")
1125
1126     s2 = time.time() #temps d'exécution de complexité en nlog(n)
1127     N_logN(len(LMod))
1128     LtempsL.append(time.time() - s2)
1129     print("Bon pour L", i+1, "fois")
1130
1131
1132 plt.plot(Ltaille,LtempsL)
1133 plt.plot(Ltaille,LtempsC)
1134 plt.legend(['Nlog(N)', 'N carré'], fontsize='xx-large')
1135 plt.title("Temps comparatif de complexité N carré et Nlog(N)", fontsize='xx-large')
1136 plt.xlabel("Taille de liste (N)", fontsize='xx-large')
1137 plt.ylabel("Temps d'exécution (s)", fontsize='xx-large')
1138 plt.show()
1139
1140 # Echelle de Mel
1141
1142 def Echelle_Mel(Hz) :
1143     '''
1144
1145     entrée : int
1146
1147     but : affiche le graphe de la perception auditive
1148     '''
1149
1150
1151     Liste_Mel = []
1152     Liste_Hz = Liste_entiers(int(Hz))
1153
1154     for x in Liste_Hz :
1155         Liste_Mel.append(Mel(x))
1156
1157
1158 plt.plot(Liste_Hz,Liste_Mel)
1159 plt.title("Perception auditive", fontsize='xx-large')
1160 plt.xlabel("Hertz (Hz)", fontsize='xx-large')
1161 plt.ylabel("Mel (M)", fontsize='xx-large')
1162 plt.show()

```