

Analyse de Fourier : 4757-6558207

Un filtre a pour fonction de transfert l'expression suivante :

$$\underline{T}(j\omega) = \frac{1}{1 + jQ \left(\frac{\omega}{\omega_0} - \frac{\omega_0}{\omega} \right)}$$

1) Préciser la nature de ce filtre en justifiant.

1. On donne ci-dessous des codes permettant de tracer les digrammes de Bode en gain et phase. Utiliser ces fonctions afin d'apprécier sur un même graphe l'influence du paramètre Q sur le gain (on prendra $Q=0.1, 1$ et 10 et $\omega_0 = 100 \text{ rad/s}$). Reprendre ce travail pour la phase.

```
In [29]: 1 import numpy as np
2 import matplotlib.pyplot as plt
3 def gain(Q,w0):
4     w=np.arange(1,100*w0,1)
5     T=(1+1j*Q*(w/w0-w0/w))**-1
6     G=np.log10(np.abs(T))
7     plt.semilogx(w,G,label=f"Q={Q}")
8     plt.grid()
9     #plt.show()
10 def phase(Q,w0):
11     w=np.arange(1,100*w0,1)
12     T=(1+1j*Q*(w/w0-w0/w))**-1
13     P=np.angle(T)
14     plt.semilogx(w,P,label=f"Q={Q}")
15     plt.grid()
16     #plt.show()
```

On envoie sur ce filtre le signal suivant :

$$e(t) = \left(\frac{1}{2} + \sum_{k=0}^{\infty} \frac{\sin((2k+1)\omega t)}{2k+1} \right)$$

- 3) On estime que le signal est correctement décrit en se limitant aux 10 première harmoniques. Proposer, en utilisant la fonction ci-dessous, une représentation graphique de ce signal en considérant les 10 premières harmoniques non nulles (on prendra $\omega=100 \text{ rad/s}$).

```
In [40]: 1 def signal(n,w):
2         periode=2*np.pi/w
3         t=np.arange(0,10*periode,periode/100)
4         e=0.5
5         for i in range(0,n):
6             e=e+np.sin((2*i+1)*w*t)/(2*i+1)
7         plt.plot(t,e)
8         plt.grid()
9         plt.xlabel("t(s)")
10        plt.ylabel("e")
11        plt.show()
12
```

4) On envoie ce signal dans le filtre précédent. On donne la fonction sortie permettant d'obtenir la représentation graphique du signal en sortie du filtre (cette fonction prendra la pulsation fondamentale w du signal et la pulsation propre w_0 comme argument ainsi que le facteur de qualité). Analyser l'effet de ce filtre pour les appels suivants :

- `sortie(100,100,0.1)`, `sortie(100,100,1)` et `sortie(100,100,10)`
- `sortie(1000,100,1)`

```
In [1]: 1 def sortie(w,w0,Q):
2         periode=2*np.pi/w
3         t=np.arange(0,10*periode,periode/100)
4         s=0
5         for i in range(0,10):
6             T=(1+1j*Q*((2*i+1)*w/w0-w0/((2*i+1)*w)))**-1
7             P=np.angle(T)
8             T=np.abs(T)
9             s=s+T*np.sin((2*i+1)*w*t+P)/(2*i+1)
10        plt.plot(t,s)
11        plt.grid()
12        plt.xlabel("t(s)")
13        plt.ylabel("s")
14        plt.show()
15
```