

1 # Détermination d'une inductance (13aa-1603540)

On souhaite déterminer l'impédance d'une bobine modélisée par une inductance L et une résistance r . On dispose pour cela d'un générateur de signaux continus et variables de résistance interne nulle et d'une boîte à décade de résistance notée R_0 .

Montage 1 : le générateur, R_0 et la bobine sont en série :

- Une première mesure est effectuée en continu. Aux bornes de $R_0 = 2\Omega$ on relève une tension deux fois plus faible que celle du générateur
- Une deuxième mesure est effectuée en régime sinusoïdal établi. A la fréquence $f=1\text{kHz}$ avec $R_0 = 10\Omega$ on relève les tensions suivantes aux bornes de R_0 et de la bobine

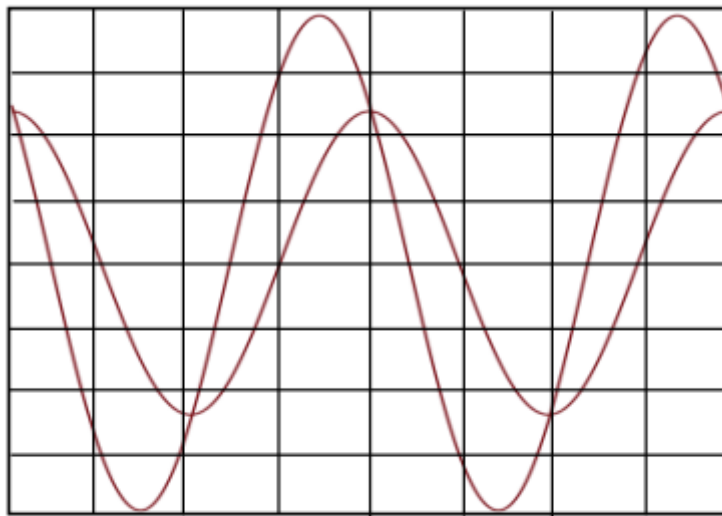


Figure 1

- 1) Faire un dessin du montage des deux mesures et préciser les difficultés expérimentales à éviter.
- 2) Déterminer les valeurs expérimentales de L et r .
- 3) Critiquer la mesure. Comment peut-on procéder autrement pour mesurer L et R ?

On utilise la bobine précédente dans le montage 2 ci-dessous où $R = 4\Omega$ et $C=1\mu\text{F}$.

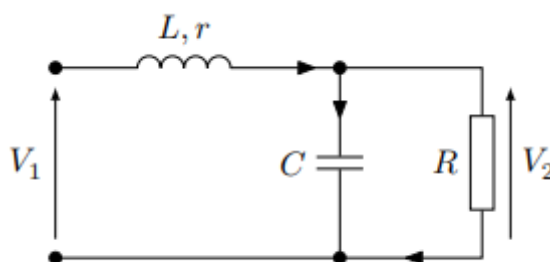


Figure 2

- 4) Obtenir la fonction de transfert $H = \frac{V_2}{V_1}$ sous la forme :

$$\underline{H} = \frac{\underline{V}_2}{\underline{V}_1} = \underline{H} = \frac{H_0}{1 + j\frac{\omega}{Q\omega_0} - \left(\frac{\omega}{\omega_0}\right)^2}$$

On exprimera Q, ω_0 et H_0 .

5) Obtenir le diagramme de Bode en gain en complétant et en utilisant la fonction `diagramme()` ci-dessous. Retrouver les propriétés caractéristiques de ce filtre sur le diagramme de Bode.

In [2]:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 def diagramme():
4     def G(w,H0,Q,w0):
5         return(20.0*np.log10(abs(H(w,H0,Q,w0))))
6     w = np.linspace(1000,1e7,10000)
7     plt.title('diagramme de Bode en gain')
8     plt.grid()
9     plt.semilogx(w,G(w,H0,Q,w0))
10    plt.show()
```

On soumet le filtre à un signal carré d'amplitude de 1V et de fréquence F_e . On donne :

- la fonction `créneau` permettant de générer ce signal carré
 - la fonction `spectre` permettant de calculer les coefficients de Fourier d'un signal périodique
 - La fonction `trace_spectre` permettant d'obtenir le spectre d'un signal de fréquence F_e
- 6) Analyser les spectres obtenus

In [12]:

```

1 import scipy.integrate as integr
2 def creneau(t):
3     if t<0: return(-1.0)
4     else: return(1.0)
5
6 def spectre(signal):
7     # coefficients de Fourier réels du signal
8     #creation d'une liste de 50 zéros
9     bnSignal = np.zeros(51)
10    for k in range(1,51):
11        #pour modifier la kème élément de liste:bnSignal[k]
12        #pour intégrer la fonction simple signal(t)*sin(2*pi*k*t) en fonction
13        # de k à chaque intégration d'où le args=(k) entre -0.5 et +0.5
14        # le [0] permet d'intégrer le calcul d'intégral dans la liste
15        bnSignal[k] = 2.0 * integr.quad(lambda t,k: signal(t)*np.sin(2*np.pi*k*t),-0.5,0.5)[0]
16
17    anSignal = np.zeros(51)
18    anSignal[0] = integr.quad(lambda t: signal(t),-0.5,0.5)[0] #valeur moyenne
19    for k in range(1,51):
20        anSignal[k] = 2.0 * integr.quad(lambda t,k: signal(t)*np.cos(2*np.pi*k*t),-0.5,0.5)[0]
21    return anSignal,bnSignal
22 def trace_spectre(signalentree,Fe):
23     #on applique la fonction spectre qui renvoie 2 valeurs renotée an et bn
24     an,bn=spectre(signalentree)
25     for k in range(0,51):
26         #tracé des raies du spectre du signal d'entrée
27         plt.plot([k*Fe,k*Fe],[0,np.sqrt((an[k])**2+(bn[k])**2)],'r',linewidth=2)
28     plt.show()
29     for k in range(0,51):
30         cn=np.sqrt((an[k])**2+(bn[k])**2)
31         #tracé du spectre du signal de sortie
32         plt.plot([k*Fe,k*Fe],[0,cn*abs(H(2*np.pi*k*Fe,H0,Q,w0))],'b',linewidth=2)
33     plt.show()
34

```