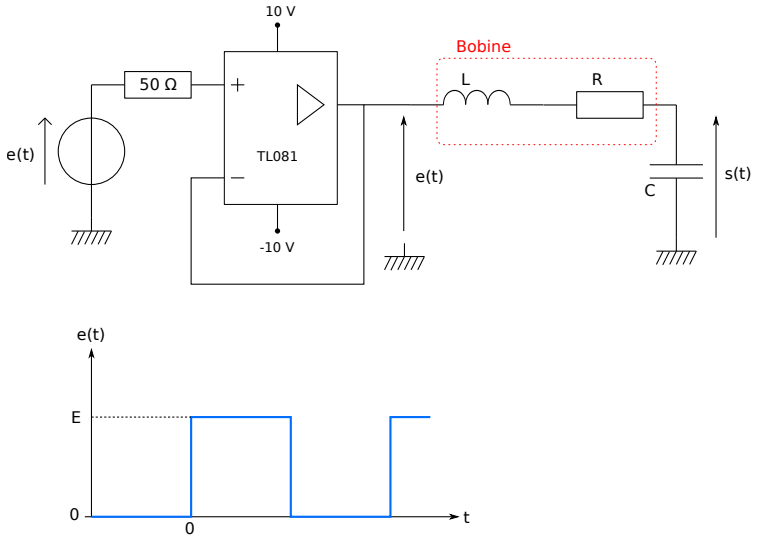
Circuit RLC : réponse à un échelon

1. Introduction

Afin de mesurer les caractéristiques d'une bobine, on étudie la réponse à un échelon (ou réponse indicielle) d'un circuit constitué de cette bobine et d'un condensateur de capacité connue.

L'analyse de la réponse permet de déterminer le coefficient d'auto-inductance de la bobine et sa résistance à la fréquence propre du circuit.

2. Montage électrique et simulation



La tension $e(t)$ appliquée au circuit RLC est fournie par un générateur de signaux dont la résistance de sortie est de $50\,\Omega$. La résistance interne d'une bobine étant de quelques ohms, il est nécessaire d'appliquer cette tension via un suiveur, réalisé au moyen de l'ALI TL081.

Supposons que, pour $t < 0$, $e = 0$ et, pour $t > 0$, $e = E$. Ce signal d'entrée constitue un échelon de hauteur E.

Pour $t > 0$, l'équation différentielle et les conditions initiales sont :

$$\frac{d^2s(t)}{dt^2} + \frac{\omega_0}{Q}\frac{ds(t)}{dt} + \omega_0^2 s(t) = \omega_0^2 E \tag{1}$$

$$\omega_0 = \frac{1}{\sqrt{LC}} \tag{2}$$

$$Q = \frac{1}{R}\sqrt{\frac{L}{C}} \tag{3}$$

$$s(0) = 0 \tag{4}$$

$$\frac{ds}{dt}(0) = 0 \tag{5}$$

Si l'on suppose que $Q > 1/2$, le temps de relaxation et la pseudo-pulsation sont définis par :

$$\tau = \frac{2Q}{\omega_0} \tag{6}$$

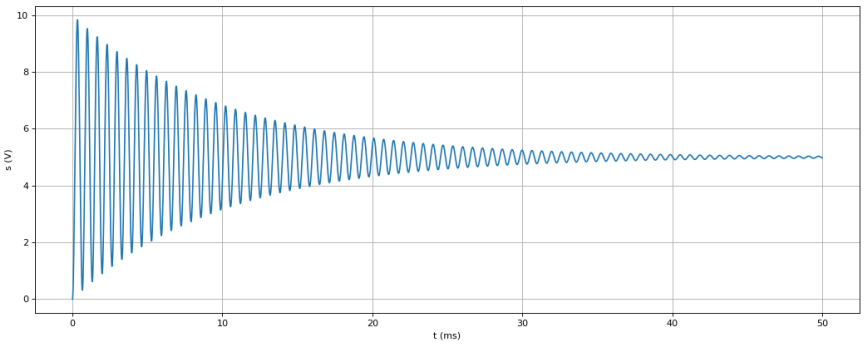
$$\omega_1 = \omega_0\sqrt{1 - \frac{1}{4Q^2}} \tag{7}$$

La solution de l'équation différentielle s'écrit :

$$s(t) = E\left(1 - e^{-\frac{t}{\tau}}\left(\cos(\omega_1 t) + \frac{1}{\tau\omega_1}\sin(\omega_1 t)\right)\right) \tag{8}$$

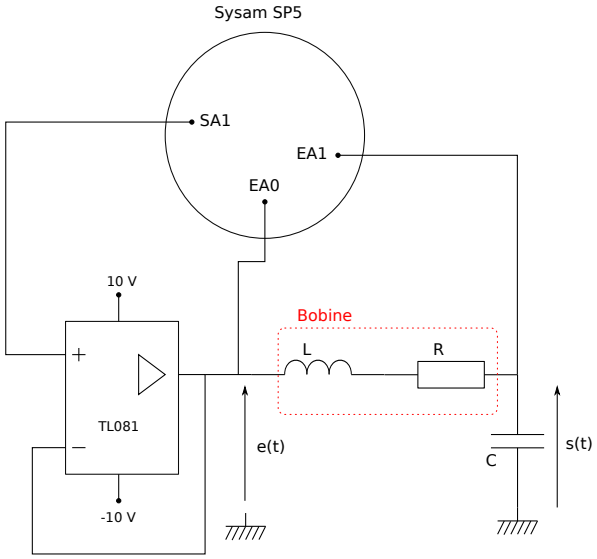
Voici un exemple :

```
import numpy as np
from matplotlib.pyplot import *

C=220e-9
L=50e-3
R=10
w0 = 1/np.sqrt(L*C)
Q = 1/R*np.sqrt(L/C)
tau = 2*Q/w0
w1 = w0*np.sqrt(1-1/(4*Q**2))
E = 5.0
t = np.linspace(0,50e-3,10000)
s = E*(1-np.exp(-t/tau)*(np.cos(w1*t)+np.sin(w1*t)/(w1*tau)))

figure(figsize=(16,6))
plot(t*1000,s)
xlabel('t (ms)')
ylabel('s (V)')
grid()
```



3. Acquisition de la réponse

La carte Sysam SP5 est utilisée pour délivrer l'échelon de tension (sortie SA1) et pour faire la numérisation des signaux $e(t)$ et $s(t)$.



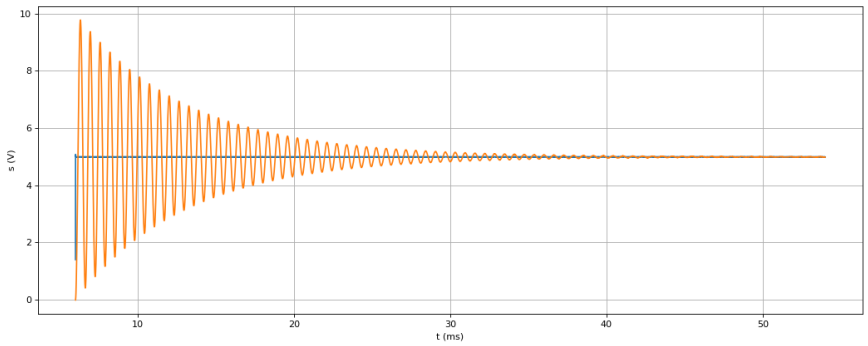
Le script ci-dessous fait l'acquisition de N points sur une durée T. La tension appliquée en sortie (SA1) est nulle pour les n_1 premiers échantillons et pour les n_1 derniers échantillons. Elle est égale à 5 volts pour les échantillons d'indices n_1 et $N - n_1$. Les tensions $e(t), s(t)$ lorsque la tension SA1 est égale à 5 volts sont enregistrées dans un fichier.

```
import numpy as np
import matplotlib.pyplot as plt
import pycanum.main as pycan

can = pycan.Sysam("SP5")
can.config_entrees([0,1],[10.0,10.0])
T=60e-3
N=10000
```

```
sortie = np.zeros(N)
n1 = N//10
sortie[n1:N-n1] = 5.0
te = T/N
can.config_echantillon(te*10**6,N)
can.acquerir_avec_sorties(sortie,sortie)
t=can.temps()[0]
signaux = can.entrees()
e=signaux[0]
s=signaux[1]
t=t[n1:N-n1]
e=e[n1:N-n1]
s=s[n1:N-n1]
np.savetxt('reponse-bobineA.txt',np.array([t,e,s]).T,fmt='%0.5e',header='t(s)\t e(V)\t s

plt.figure()
plt.plot(t,e)
plt.plot(t,s)
plt.grid()
plt.show()
```

Voici le signal obtenu pour une bobine de 1000 spires (sans noyau) :

```
[t,e,s] = np.loadtxt('reponse-bobineA.txt',skiprows=1,unpack=True)
figure(figsize=(16,6))
plot(t*1000,e)
plot(t*1000,s)
xlabel('t (ms)')
ylabel('s (V)')
grid()
```

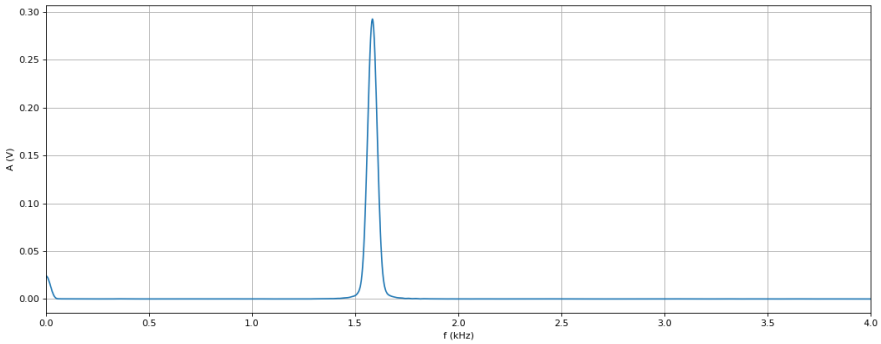


4. Traitement du signal

L'objectif est de déterminer la pulsation ω_1 et le temps de relaxation τ.

Nous effectuons tout d'abord une analyse spectrale du signal (retranché de sa valeur moyenne) par transformée de Fourier discrète. La durée du signal analysé n'est pas très grande devant la pseudo-période. Il faut donc effectuer une interpolation du spectre en complétant le signal par des zéros après lui avoir appliqué une fenêtre (Blackman).

```
from numpy.fft import fft
from scipy.signal import blackman
E = 5.0
s = s-E
N = len(s)
x = np.concatenate((s*blackman(N),np.zeros(6*N)))
N1=len(x)
spectre = np.absolute(fft(x))*2.0/N/0.42
Te = t[1]-t[0]
freq=np.arange(N1)*1/(N1*Te)
i = np.argmax(spectre[0:N1//2])
f1 = freq[i]


figure(figsize=(16,6))
plot(freq/1e3,spectre)
xlabel('f (kHz)')
ylabel('A (V)')
xlim(0,4)
grid()
```



Voici la fréquence obtenue :

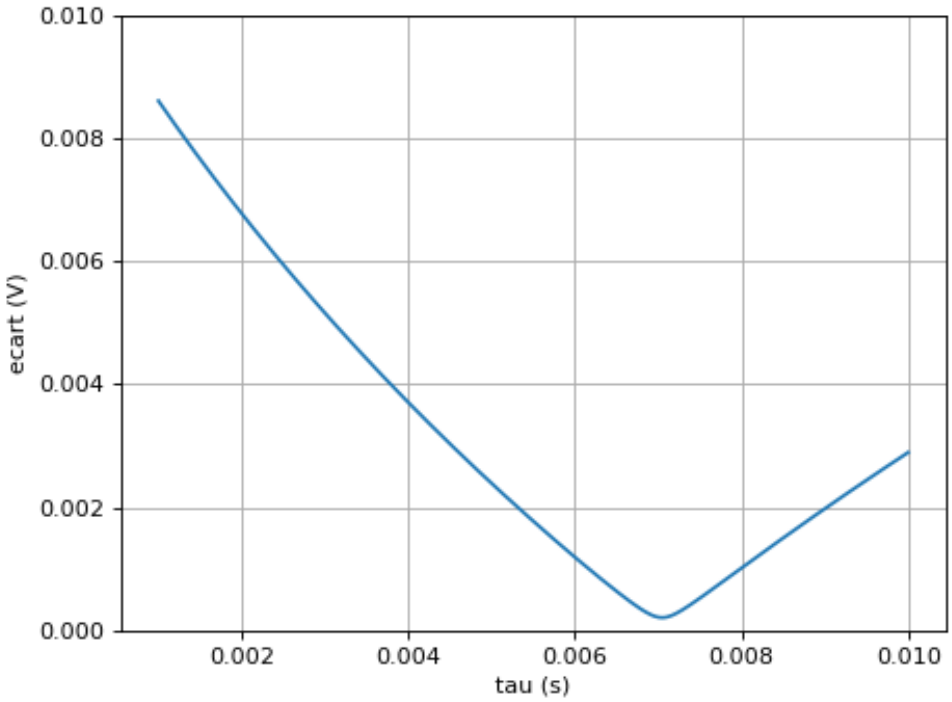
```
print(f1)
--> 1583.3333333333521
```

Afin de déterminer le temps de relaxation, nous écrivons une fonction qui calcule l'écart quadratique entre le signal expérimental et la forme théorique de $s(t)$ (retranchée de E).

```
def ecart(s,E,te,f1,tau):
    N = len(s)
    w1 = 2*np.pi*f1
    t = np.linspace(0,te*N,N)
    y = s+E*np.exp(-t/tau)*(np.cos(w1*t)+1/(tau*w1)*np.sin(w1*t))
    return np.sqrt(np.sum(y**2))/N
```

Voici un tracé de cet écart en fonction du temps de relaxation :

```
P = 1000
tau_min = 1e-3
tau_max = 10e-3
tau = np.linspace(tau_min,tau_max,P)
ec = np.zeros(P)
for i in range(P):
    ec[i] = ecart(s,5,Te,f1,tau[i])
tau_opt = tau[np.argmin(ec)]
ec_min = ec.min()

figure()
plot(tau,ec)
xlabel('tau (s)')
ylabel('ecart (V)')
ylim(0,1e-2)
grid()
```



Voici la valeur optimale du temps de relaxation, correspondant au minimum de l'écart :

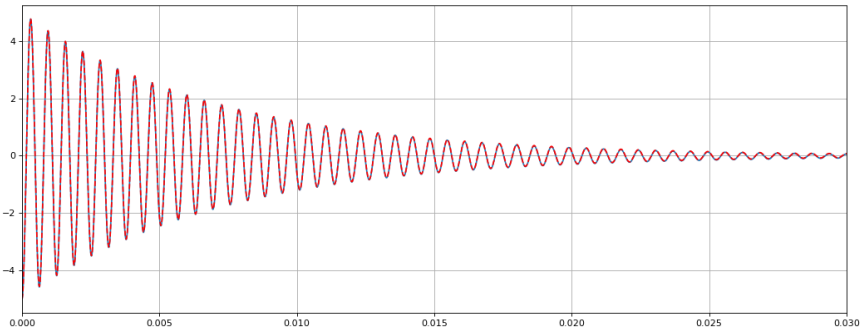
```
print(tau_opt)
--> 0.007045045045045046
```

et voici la valeur minimale :

```
print(ec_min)
--> 0.000205320332094271
```

Pour finir, on trace le signal expérimental et la courbe du modèle :

```
N = len(s)
t = np.linspace(0,Te*N,N)
w1 = 2*np.pi*f1
y = -E*np.exp(-t/tau_opt)*(np.cos(w1*t)+1/(tau_opt*w1)*np.sin(w1*t))
figure(figsize=(16,6))
plot(t,s)
plot(t,y,'r--')
xlim(0,0.03)
grid()
```



5. Caractéristiques de la bobine

Le très bon accord entre la courbe expérimentale est celle du modèle montre que la représentation de cette bobine par L et R en série est bonne.

Voici les relations qui permettent de calculer ces coefficients :

$$\omega_0^2 = \omega_1^2 + \frac{1}{\tau^2} \tag{9}$$

$$L = \frac{1}{C\omega_0^2} \tag{10}$$

$$R = \frac{2L}{\tau} \tag{11}$$

La capacité du condensateur est mesurée avec un RLC-mètre : $C = 223{,}4\,\mathrm{nF}$ (valeur mesurée à la fréquence 1 kHz). Nous considérons que l'incertitude de cette valeur est de 1 nF.

```
C=223.4e-9
delta_C = 1e-9
w1=2*np.pi*f1
L=1/(C*(w1**2+1/tau_opt**2))
R=2*L/tau_opt
```

```
print(L)
--> 0.04521931523661002
```

```
print(R)
--> 12.837196908602843
```

L'incertitude de la fréquence f_1 est principalement due à la précision de l'analyse spectrale. Elle est donc égale à la moitié de l'inverse de la durée d'analyse :

```
delta_f1 = 1/(len(s)*Te)
```

L'incertitude du temps de relaxation est donnée par la précision de l'échantillonnage de l'écart :

```
delta_tau = (tau_max-tau_min)/P
```

Si l'on prend en compte seulement ces deux incertitudes, les incertitude-types de L et C peuvent être obtenues par une méthode de Monte-Carlo, qui consiste à simuler les variations aléatoires des valeurs mesurées de f_1 et de τ, et des valeurs de C, au moyen de nombres aléatoires tirés avec la loi de probabilité normale :

```
import numpy.random as random
Nt = 100000
tirages_f1 = random.normal(f1,delta_f1,Nt)
tirages_tau = random.normal(tau_opt,delta_tau,Nt)
tirages_C = random.normal(C,delta_C,Nt)
tirages_w1=2*np.pi*tirages_f1
tirages_L=1/(tirages_C*(tirages_w1**2+1/tirages_tau**2))
tirages_R=2*tirages_L/tirages_tau
delta_L = tirages_L.std() # écart-type des tirages
delta_R = tirages_R.std()
```

```
print(delta_L)
--> 0.0012082788152699

print(delta_R)
--> 0.34345215778214605
```

Nous obtenons donc :

$$L = 45{,}2 \pm 1{,}2\,\mathrm{mH}$$
$$R = 12{,}8 \pm 0{,}3\,\Omega$$

Il faut préciser que la valeur de R obtenue prend en compte toute les pertes (y compris dans le condensateur) à la fréquence d'oscillation. Voici, pour la même bobine, les résultats obtenus avec un condensateur de capacité $1\,\mu$F :

```
f1 = 752.232143 Hz
R = 10.617984 +/- 0.442298 ohms
L = 0.045198 +/- 0.001882 H
```

La valeur du coefficient d'auto-inductance est toujours la même, mais celle de la résistance est plus petite parce que la fréquence d'oscillation est plus basse, ce qui implique un moindre effet de peau. Si l'objectif est de modéliser la bobine afin de l'intégrer dans un circuit résonant, il faudra faire la mesure avec le condensateur prévu pour l'application. On aura ainsi une estimation correcte de la résistance du circuit résonant alors qu'une mesure avec un ohmmètre donne une valeur sous-estimée.

6. Script d'analyse

Le script analyseBobine.py effectue toutes les opérations décrites précédemment. La durée de l'analyse et la valeur de la capacité sont définies au début du script. Lorsque L est plus faible, on augmente C de manière à maintenir une fréquence d'oscillation de l'ordre de $1\,\mathrm{kHz}$. L'emploi d'un condensateur polarisé est possible puisque la tension en sortie est positive.