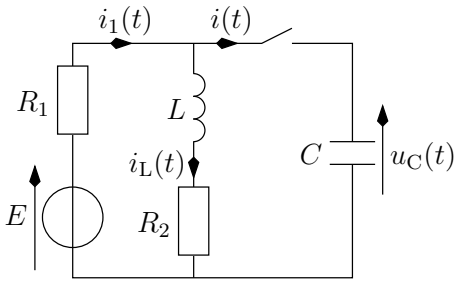
DM n° 6 de Physique
Régime transitoire d'ordre 2

Étude de la charge d'un condensateur

On considère le circuit ci-contre et on souhaite y étudier l'évolution transitoire de la tension u_C. Au début de l'expérience que l'on modélise, le condensateur est chargé à la tension $u_C = u_0$ et l'interrupteur est ouvert *depuis très longtemps*.



A Valeurs initiales et finales

On souhaite dans un premier temps établir les valeurs initiales et finales des courants et tensions.

1. À l'aide d'un nouveau schéma, déterminer la valeur des courants $i(t=0^-)$, $i_L(t=0^-)$ et $i_1(t=0^-)$ avant la fermeture de l'interrupteur.
2. En déduire la valeur de $u_C(t=0^+)$, $i_1(t=0^+)$, $i_L(t=0^+)$ et $i(t=0^+)$ après la fermeture de l'interrupteur.
3. À l'aide d'un schéma supplémentaire, déterminer la valeur de la tension $u_C(t=\infty)$ en régime permanent, au bout d'un temps très long.

B Équation différentielle

On cherche désormais à établir l'équation différentielle régissant l'évolution de la tension u_C.

4. Donner les quatre équations électriques reliant i_1, i_L, i et u_C. Ces équations peuvent faire apparaître les dérivées de ces fonctions et bien sûr les données du problème (E, R_1, R_2, L et C).
5. À partir d'une de ces équations, exprimer i_1 en fonction u_C et des données du problème.
6. À partir de deux équations parmi celles de la question 4, établir une relation entre i_1, i, $\dfrac{\mathrm{d}i}{\mathrm{d}t}$ et $\dfrac{\mathrm{d}i_1}{\mathrm{d}t}$.
7. Déduire des deux questions précédentes une relation reliant u_C, i et leurs dérivées, faisant intervenir E, R_1, R_2 et L.
8. En déduire finalement l'équation différentielle régissant l'évolution de la tension u_C.
9. Montrer que la pulsation propre ω_0 et le facteur de qualité Q du système s'écrivent

$$\omega_0 = \sqrt{\frac{R_1 + R_2}{R_1\,L\,C}} \qquad \text{et} \qquad Q = \frac{\sqrt{R_1\,(R_1+R_2)\,L\,C}}{L + R_1\,R_2\,C}$$

C Résolution de l'équation différentielle

On choisit $R_1 = 470\,\Omega$, $R_2 = 10\,\Omega$, $L = 8\,\mathrm{mH}$, $C = 2200\,\mathrm{nF}$, $E = 20\,\mathrm{V}$ et $u_0 = 1\,\mathrm{V}$.

10. Application numérique : calculer Q. De quel régime s'agit-il ?
11. En déduire l'expression complète de $u_C(t)$, sans déterminer les constantes d'intégration.
12. Exprimer les deux conditions initiales $u_C(0)$ et $\dot{u}_C(0)$ et les calculer (en V et en V/ms).
13. Tracer qualitativement l'évolution de $u_C(t)$. Il faut pour cela calculer numériquement la constante de temps de l'enveloppe exponentielle, la valeur finale, et la pseudo-période.
14. Déterminer les constantes d'intégration. Conseil : utiliser une forme $A\cos(\omega_p\,t) + B\sin(\omega_p\,t)$ pour la partie pseudo-périodique et noter $\dot{u}_C(0)$ pour ces calculs, que l'on remplacera au dernier moment.

D Cadeau bonus

Voilà le script qui trace $u(t)$. Mais il faut le télécharger et le compléter pour avoir droit à la figure !

```
# Importation des modules nécessaires
import numpy as np
import matplotlib.pyplot as plt

# Script à compléter. Inutile de le lancer sans avoir complété !

# Données
E =
u0 =
r1 =
r2 =
l = 8e-3 # Éviter 8*10**(-3), bien moins précis. 8e-3 est un simple nombre
c =

# Constantes définies dans le sujet et permettant de réaliser le tracé
Q = np.sqrt()/()
omega0 = ...
tau = 2*Q/omega0
omegap = omega0/(2*Q) * np.sqrt(...)
up0 = 1/(r1*c) * ( r2/(r1+r2)*E - u0 )
ufinal = ...*E
# Affichage
print("Q      =",round(Q,2))
print("omega0 =",int(omega0),'rad/s')
print("tau    =",round(1000*tau,3),'ms')
print("omegap =",int(omegap),'rad/s')
print("taup   =",round(1000*.../omegap,3),'ms')
print("up0    =",round(up0/1000,3),'V/ms')
print("ufinal =",round(ufinal,3),'V')

# Constantes d'intégration
A = u0-ufinal
B = ( up0 + A/tau ) / omegap

# Fonctions temporelles
def enveloppe(t):
    return np.exp(-t/tau)
def u(t):
    return ufinal + enveloppe(t) * ( A*np.cos(omegap*t) + B*np.sin(...) )

# Tracé de l'évolution du système
t = np.linspace(0, ..., 2001) # de 0 à 5*tau, 2001 points : dt = 5*tau/2000
plt.plot(1000*t, ...) # temps en abscisses en millisecondes
plt.xlabel('Temps (ms)')
plt.ylabel('Tension u_c (V)')
plt.grid()
# Enveloppe exponentielle
plt.plot(1000*t, ufinal + np.sqrt(A**2+B**2) * enveloppe(t), 'r--')
plt.plot(..., ufinal - np.sqrt(A**2+B**2) * ..., 'r--')
plt.show()
```