Étude d'un circuit RLC en régime libre et forcé

I – Introduction

L'objectif de ce micro-projet informatique est la résolution numérique de l'équation différentielle du second ordre, d'un système en régime libre ou forcé. Le système que j'ai étudié est le circuit RLC (résistance bobine condensateur) série en régime libre, puis soumis à une excitation sinusoïdale.

Présentation du problème physique : L'évolution du circuit RLC série soumis à une tension e(t) est gouvernée par l'équation différentielle du second ordre suivante :

$$\ddot{q} + \frac{\omega_0}{Q}\dot{q} + {\omega_0}^2 q = F(t)$$

Pour toute la suite, on a $\omega_0 = \frac{1}{\sqrt{LC}}$ et $Q = \frac{L\omega_0}{R} = \frac{1}{R}\frac{\sqrt{L}}{C} = \frac{1}{RC\omega_0}$, et on choisit la valeur des paramètres ω_0=2 et Q=2.

On souhaite mettre en évidence l'impact du pas de discrétisation sur la qualité des résultats et sur le temps de calcul. On comparera les résultats obtenus avec ceux des fonctions déjà implémentées dans la bibliothèque numérique Scipy de Python, et avec des résultats expérimentaux. Le plan d'étude est le suivant :

1°) Résolution de l'équation du système par la méthode d'Euler

a) En régime libre

b) En régime forcé

2°) Résolution de l'équation du système grâce à des fonctions de Scipy

a) En régime libre

b) En régime forcé

II - Description succincte des algorithmes

1°) Programme de résolution utilisant la méthode d'Euler

Soit l'équation du circuit RLC : $\ddot{q} + (\frac{\omega_0}{Q})\dot{q} + (\omega_0)^2 q = f(t)$, par abus de notation, on notera f(t)=f.

On obtient :

$$\ddot{q} + (\frac{\omega_0}{Q})\dot{q} + (\omega_0)^2 q - f = 0$$

On pose $Y=\begin{pmatrix} q \\ \dot{q} \end{pmatrix}$ on a donc $\dot{Y}=\begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix}=\begin{pmatrix} \dot{q} \\ \frac{-\omega_0}{Q}\dot{q} - (\omega_0)^2 q + f \end{pmatrix}$. On pose $z = \dot{q}$ donc

$$\dot{Y}=\begin{pmatrix} z = \dot{q} \\ \dot{z} = \frac{-\omega_0}{Q} z - (\omega_0)^2 q + f \end{pmatrix}$$

La relation d'Euler s'écrit alors :

$$q_{k+1} = q_k + hz_k$$
$$z_{k+1} = z_k + h(\frac{-\omega_0}{Q} z - (\omega_0)^2 q + f)$$

a) En régime libre : on pose f(t)=0 et on suppose que q(0)=q_0 et $\dot{q}(0) = 0$

La relation d'Euler est alors :

$$q_{k+1} = q_k + hz_k$$
$$z_{k+1} = z_k + h(\frac{-\omega_0}{Q} z - (\omega_0)^2 q)$$

On pose pour le programme q_0=q et z_0=0.

b) En régime forcé : on pose f(t)= A*cos(ωt) et on suppose que q(0)=q_0 et que $\dot{q}(0) = 0$. La relation d'Euler est alors :

$$q_{k+1} = q_k + hz_k$$
$$z_{k+1} = z_k + h(\frac{-\omega_0}{Q} z - (\omega_0)^2 q + A * \cos(\omega t))$$

Programmes : (Voir figure 1 a) et b) en annexe)

Graphiques obtenus : (Voir figure 2 a) et b) en annexe)

2°) Programme utilisant la fonction déjà implémentée « odeint »

Notations : $\mathrm{q}_{dot} = \dot{\mathrm{q}} = \mathrm{z}$ et $\mathrm{q}_{dot2} = \ddot{\mathrm{q}} = \dot{\mathrm{z}}$.

Le but de ce programme est de résoudre l'équation différentielle du circuit RLC série, afin de déterminer l'évolution temporelle de la charge q du condensateur dans le circuit.

Pour cela nous utilisons la fonction « odeint » qui permet de résoudre une équation différentielle d'ordre 1. Or l'équation différentielle régissant le circuit RLC est d'ordre 2, c'est pourquoi nous avons introduit une fonction f qui permet de « transformer » notre équation d'ordre 2 en ordre 1.

Cette fonction renvoie un vecteur de 2 lignes et de n colonnes ayant pour coordonnées : $\mathrm{Y}' = \begin{pmatrix} \dot{\mathrm{q}} \\ \ddot{\mathrm{q}} \end{pmatrix} = \begin{pmatrix} z \\ \dot{\mathrm{z}} \end{pmatrix}$, $\dot{z}$ étant obtenu grâce à l'équation différentielle : $\ddot{\mathrm{q}} = -\left(\frac{\omega_0}{\mathrm{Q}}\right)\dot{\mathrm{q}} - (\omega_0)^2\mathrm{q}$ pour le régime libre, et $\ddot{\mathrm{q}} = -\left(\frac{\omega_0}{\mathrm{Q}}\right)\dot{\mathrm{q}} - (\omega_0)^2\mathrm{q} + \mathrm{f}$ pour le régime forcé.

Lorsque nous intégrons la fonction f à l'aide d'odeint (après avoir posé les conditions initiales $\mathrm{Y}_0 = \begin{pmatrix} q_0 \\ 0 \end{pmatrix}$ avec q_0=2) nous obtenons le vecteur : $\mathrm{Y} = \begin{pmatrix} q \\ \dot{q} \end{pmatrix} = sol_{\mathrm{ode}}$, comme $\mathrm{Y}[0] = q = \mathrm{sol}_{\mathrm{ode}}[:,0]$ il ne nous reste plus qu'à tracer le graphique de coordonnées (t,q(t)) à l'aide de « matplotlib.pyplot ».

Pour le graphique, nous avons créé une liste de temps de 0 seconde à 40 secondes à l'aide de la fonction « linspace ».

Programmes : (Voir figure 3 a) et b) en annexe)

Graphiques obtenus : (Voir figure 4 a) et b) en annexe)

III - Les différentes études et analyses

Dans le cadre de la résolution de l'équation différentielle en régime libre par la méthode d'Euler, la valeur initiale de Q influe sur l'amortissement de la courbe, en effet plus la valeur de Q est élevée, plus l'amortissement est faible voir nul.

Voir les figures 5 a) b) et c) : ayant respectivement pour valeurs Q=2, Q=4 et Q=8, en annexe.

Les variations du pas h n'ont pas influé sur le résultat des simulations, en effet en faisant varier le pas entre 0,01 et 1 je n'ai pas pu remarquer de variation sur les courbes pour la méthode d'Euler ainsi que celle d'Odeint. Cependant, on sait que théoriquement plus h diminue (donc n est grand) plus le résultat est censé être précis.

J'ai ensuite superposé les courbes obtenues par la méthode d'Euler et par celle d'Odeint, afin de comparer et de noter les écarts entre les résultats des deux méthodes. Les valeurs choisies sont : q_0=2, l'amplitude vaut A=4, la pulsation ω=1, le facteur de qualité Q=8, et enfin la pulsation propre ω_0=2.

Graphiques obtenus : voir figures 6 a) b) et c) en annexe.

Les résultats montrent que dans les deux régimes, les deux méthodes fournissent des résultats différents en termes d'amplitude. En effet, on remarque que la courbe de la méthode d'Euler a une amplitude plus grande dans les deux régimes. Cependant, il y a une importante différence au niveau de la période des oscillations dans le cas du régime forcé. En effet, la courbe représentant la méthode d'Odeint semble posséder une période cinq fois supérieure à celle d'Euler. Nous pouvons émettre l'hypothèse d'une erreur de programmation pour les valeurs Q ou ω_0.

Conclusion

Ce projet m'a permis de résoudre une équation différentielle importante du programme de physique de cette année. En effet le circuit RLC et ses applications sont une partie essentielle dans l'étude des circuits électroniques. J'ai utilisé l'outil numérique afin de résoudre pour différentes conditions initiales cette équation difficile à résoudre normalement. Les résultats des simulations ont montré que, par la méthode d'Euler comme par la méthode d'odeint, il est possible d'étudier l'importance de certains paramètres dans un circuit. Les graphiques obtenus ont permis d'étudier les variations de la charge q au cours du temps, et de comparer les deux méthodes numériques que j'ai utilisées.

Annexes

Programmation de la méthode d'Euler : figure 1 a) : Régime libre

```
def EQD_libre(t0,tf,j,p,q0): #t0 et tf designent respectivement le temps
#initial et le temps final, j designe w0, et p designe Q.
    h=0.05 #h designe le pas
    t,q,z=t0,q0,0 #valeurs initiales
    T,Q,Z=[t],[q],[z] #création de 3 listes
    while t<tf:
        t,q,z=t+h,q+h*z,z+h*((-j*z)/p-q*j**2)
        T.append(t)
        Q.append(q)
        Z.append(z)
    return T,Q

def graphe_libre(t0,tf,j,p,q0): #création d'une fonction qui permet de tracer
#q(t) en éxecutant la fonction précédente
    T,Q=EQD_libre(t0,tf,j,p,q0)
    #création du graphique
    plt.plot(T,Q,label='blue')
    plt.legend(loc="upper center")
    plt.title("Modélisation du circuit RLC par la méthode d'Euler")
    plt.xlabel("temps (seconde)") #abscisse
    plt.ylabel("Q(t)") #ordonnée
    plt.legend('C')
    plt.grid()
    plt.show()
```

Figure 1 b) : Régime forcé

```
    #Résolution par la méthode d'Euler : Régime forcé

    def EQD_force(t0,tf,j,p,q0,A,w):
        h=0.05 #h designe le pas
        t,q,z=t0,q0,0 #valeurs initiales
        T,Q,Z=[t],[q],[z] #création de 3 listes
        while t<tf:
            t,q,z=t+h,q+h*z,z+h*((-j*z)/p-q*j**2+A*np.cos(w*t))
            T.append(t)
            Q.append(q)
            Z.append(z)
        return T,Q


    def graphe_force(t0,tf,j,p,q0,A,w): #création d'une fonction qui permet de
    #tracer q(t) en executant la fonction précédente
        T,Q=EQD_force(t0,tf,j,p,q0,A,w)
        #création du graphique
        plt.plot(T,Q,label='blue')
        plt.legend(loc="upper center")
        plt.title("Modélisation du circuit RLC par la méthode d'Euler")
        plt.xlabel("temps (seconde)") #abscisse
        plt.ylabel("Q(t)") #ordonnée
        plt.legend('C')
        plt.grid()
        plt.show()
```

Graphiques obtenus : figure 2 a) : Régime libre

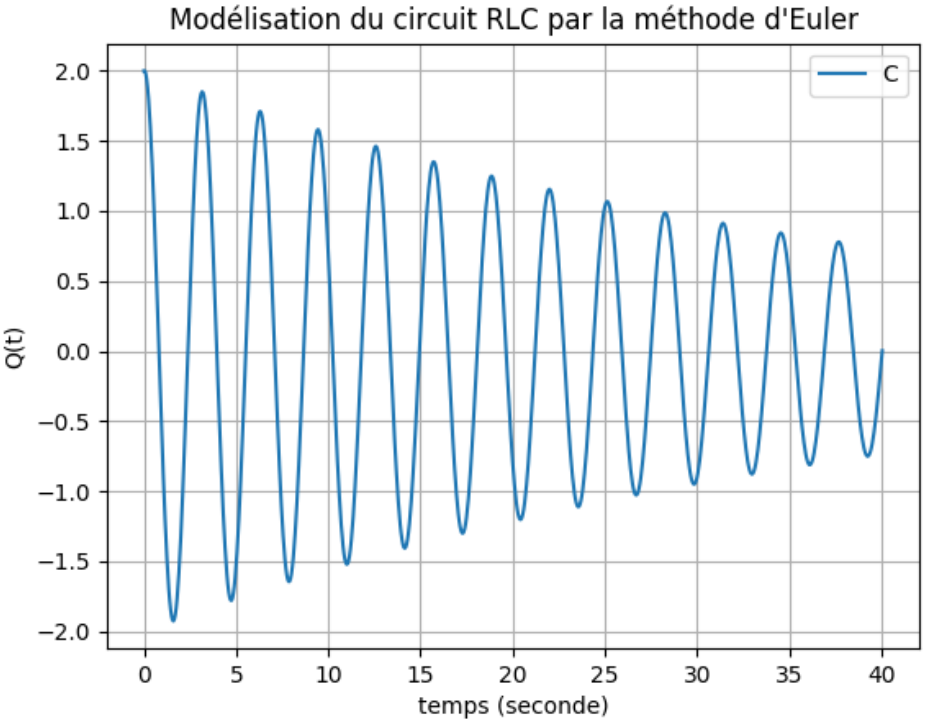
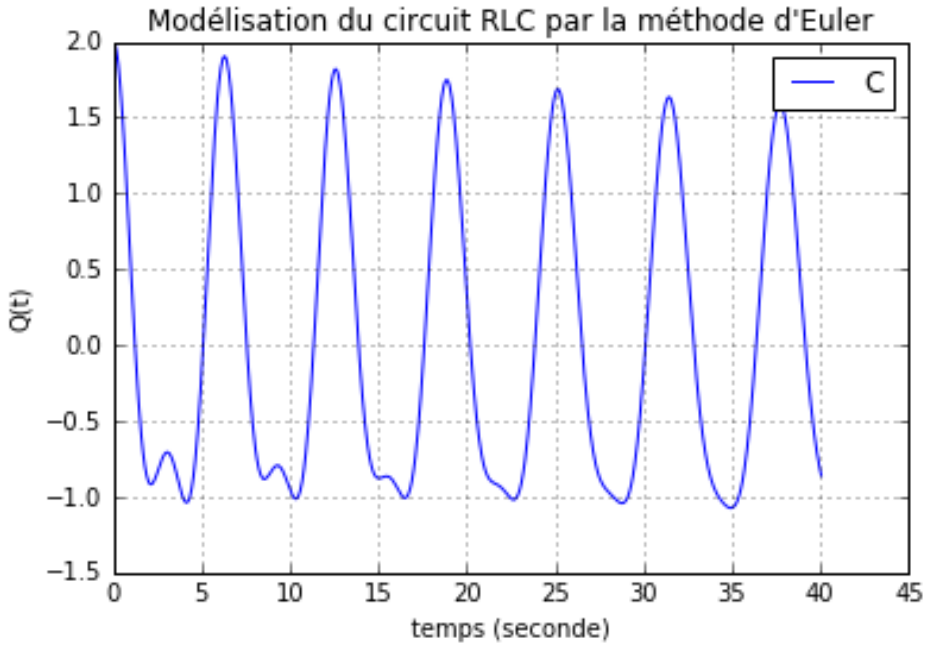


Figure 2 b) : Régime forcé



Programmation de la méthode d'Odeint : figure 3 a) : Régime libre

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint

#z = qdot
#w0=10, Q=5,l'amplittude vaut A=4, et la pulsation w=1

t = np.linspace (0,40,1000)


def f(q,t,w0,Q):#q_dot=f(q,t)

    z_dot = -(w0/Q)*q[1]-w0**2*q[0] #q[1]=q_dot et q[0]=q
    return [q[1],z_dot] #vecteur (q_dot,q_dot2)


sol_ode = odeint(f,(2,0),t,args=(5,100))#2=q0 et args=(w0/Q,w0**2)
plt.title("Evolution temporelle de q(t) en régime libre")
plt.plot (t,sol_ode[:,0],"b")
plt.xlabel('Temps(s)')
plt.ylabel('Charge q')
plt.grid()
plt.figure()
plt.show()
plt.savefig("figure.pdf")
```

Figure 3 b) : Régime forcé

```
def F(q,t,w0,Q):#q_dot=f(q,t)

    z_dot = 4*np.cos(1*t)-(w0/Q)*q[1]-w0**2*q[0] #q[1]=q_dot et q[0]=q
    return [q[1],z_dot] #vecteur (q_dot,q_dot2)


sol_ode = odeint(F,(2,0),t,args=(5,100))#2=q0 et args=(w0/Q,w0**2)
plt.title("Evolution temporelle de q(t) en régime forcé")
plt.plot (t,sol_ode[:,0],"b")
plt.xlabel('Temps(s)')
plt.ylabel('Charge q')
plt.grid()
plt.figure()
plt.show()
plt.savefig("figure.pdf")
```

Graphiques obtenus : figure 4 a) : Régime libre

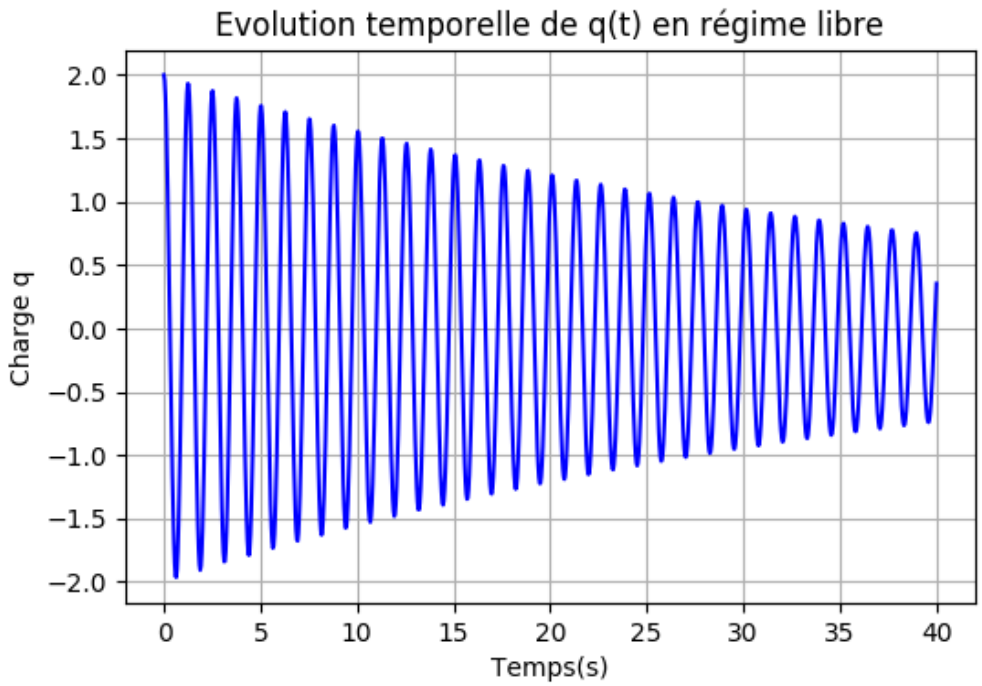
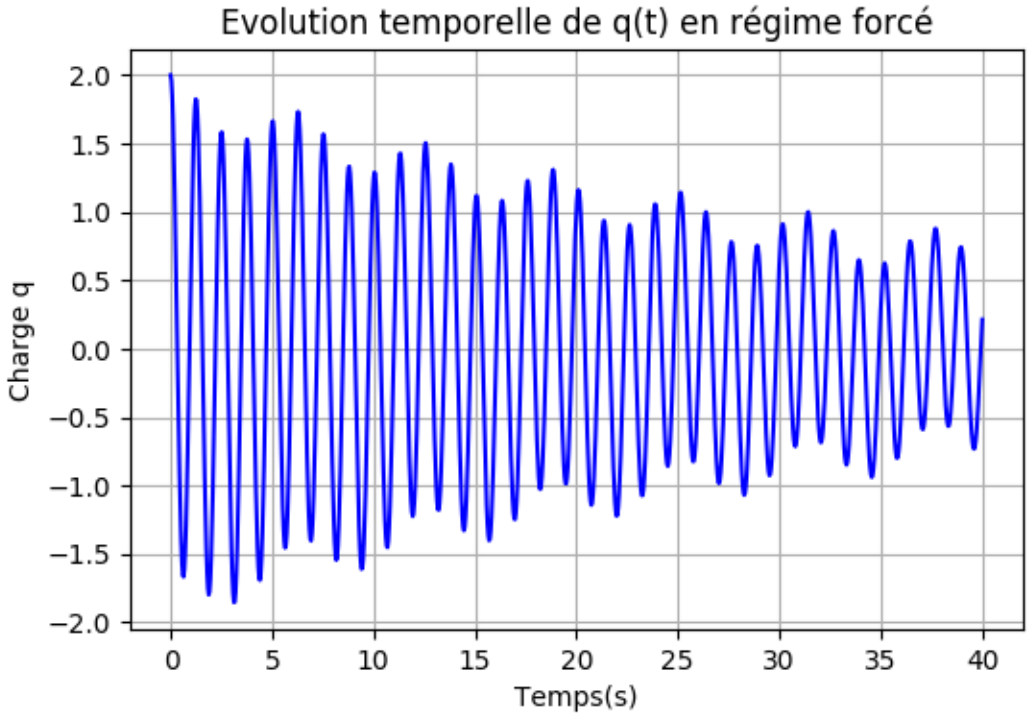
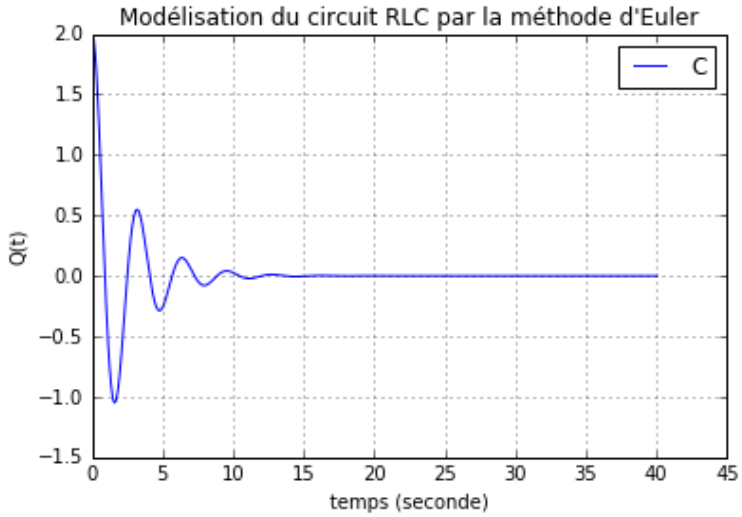


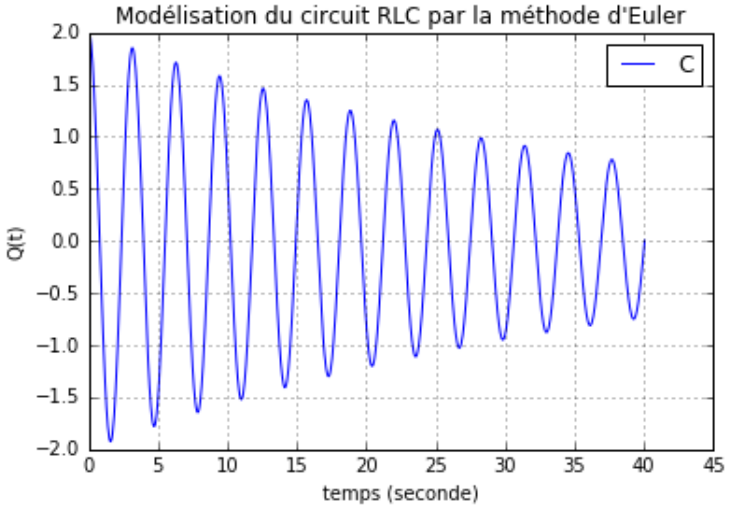
Figure 4 b) : Régime forcé

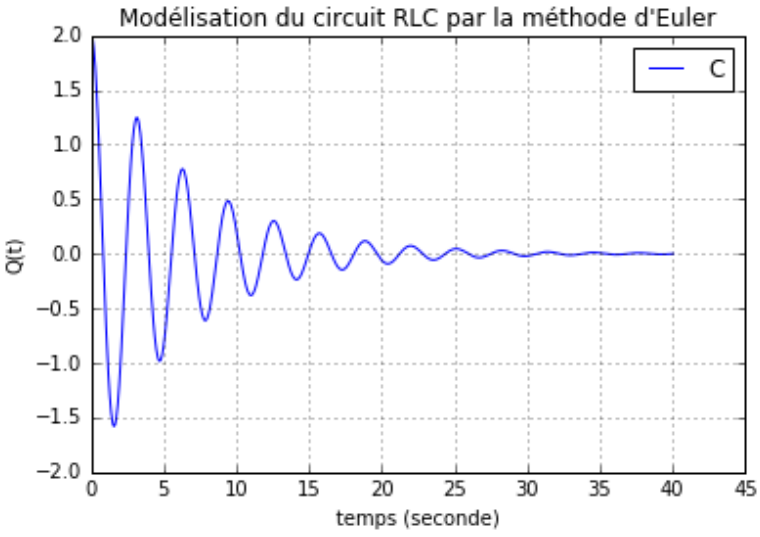


Analyse des résultats :

Figures 5 a) b) et c), ayant respectivement pour valeurs Q=2, Q=4 et Q=8 :







Superposition des courbes obtenues par la méthode d'Euler et d'Odeint :

Figure 6 a) : En régime libre

```
def graphe_libre(t0,tf,j,p,q0):
    T,Q=EQD_libre(t0,tf,j,p,q0) #solution d'Euler
    #création du graphique
    plt.plot(T,Q,'b',label='Euler')
    plt.plot (t,sol_ode[:,0],"r",label='Scipy') #sol_ode est la solution d'Odeint
    plt.legend(loc="upper center")
    plt.title("Modélisation du circuit RLC par la méthode d'Euler et d'Odeint")
    plt.xlabel("temps (seconde)") #abscisse
    plt.ylabel("Q(t)") #ordonnée
    plt.legend()
    plt.grid()
    plt.show()
```

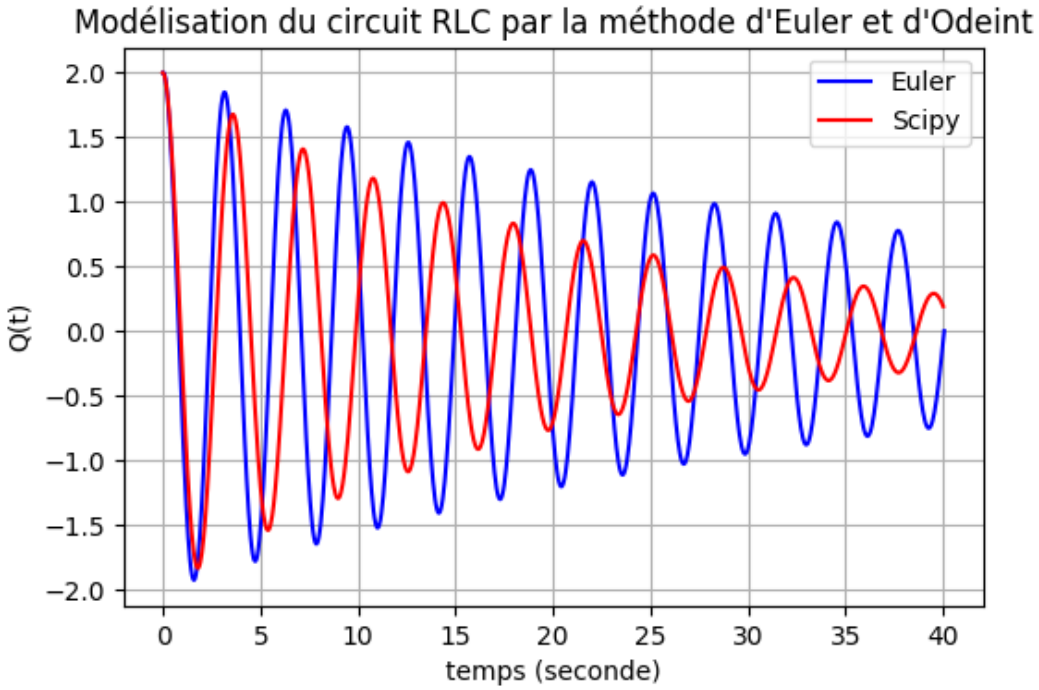


Figure 6 b) : En régime forcé

```
def graphe_force(t0,tf,j,p,q0,A,w):
    T,Q=EQD_force(t0,tf,j,p,q0,A,w) #solution d'Euler
    #création du graphique
    plt.plot(T,Q,'b', label='Euler')
    plt.plot (t,sol_ode[:,0],"r",label='Scipy') #sol_ode est la solution d'Odeint
    plt.legend(loc="upper center")
    plt.title("Modélisation du circuit RLC par la méthode d'Euler et d'Odeint")
    plt.xlabel("temps (seconde)") #abscisse
    plt.ylabel("Q(t)") #ordonnée
    plt.legend()
    plt.grid()
    plt.show()
```