

TP RÉSOLUTION D'ÉQUATION DIFFÉRENTIELLE VECTORIALISATION

Travail préparatoire : bien revoir le TP sur la résolution d'une équation différentielle d'ordre 1 (théorie + scripts dans Capytale) + lire les pages 1 à 3 de ce TP + répondre aux questions du 1.4 et du 2.2

1 Rappels sur la résolution d'une équation différentielle d'ordre 1

1.1 La théorie

Depuis le TP sur la résolution d'une équation différentielle d'ordre 1, nous savons résoudre une équation différentielle écrite sous la forme $y' = F(y, t)$, connaissant la condition initiale $y(t_0) = y_0$ (problème appelé « problème de Cauchy »).

Pour cela, on utilise **la méthode d'Euler** qui consiste à choisir un pas Δt afin de calculer les valeurs approchées de la fonction $y(t)$, solution de l'équation différentielle, à différentes date $t_i = t_0 + i \times \Delta t$, en utilisant le fait que :

$$y'(t_i) = \frac{dy}{dt}(t_i) \approx \frac{y_{i+1} - y_i}{\Delta t}, \text{ ce qui équivaut à : } y_{i+1} \approx y_i + \frac{dy}{dt}(t_i) \times \Delta t$$

On obtient donc :

$$\begin{cases} y(t = t_0) = y_0 \\ y_1 = y(t_1 = t_0 + \Delta t) = y_0 + \frac{dy}{dt}(t_0) \times \Delta t = y_0 + F(y_0, t_0) \times \Delta t \\ y_2 = y(t_2 = t_1 + \Delta t) = y_1 + \frac{dy}{dt}(t_1) \times \Delta t = y_1 + F(y_1, t_1) \times \Delta t \\ \dots \\ y_{i+1} = y(t_{i+1} = t_i + \Delta t) = y_i + \frac{dy}{dt}(t_i) \times \Delta t = y_i + F(y_i, t_i) \times \Delta t \end{cases}$$

Ce qui permet alors de calculer numériquement et « de proche en proche » des valeurs approchées de la solution $y(t)$ de l'équation différentielle.

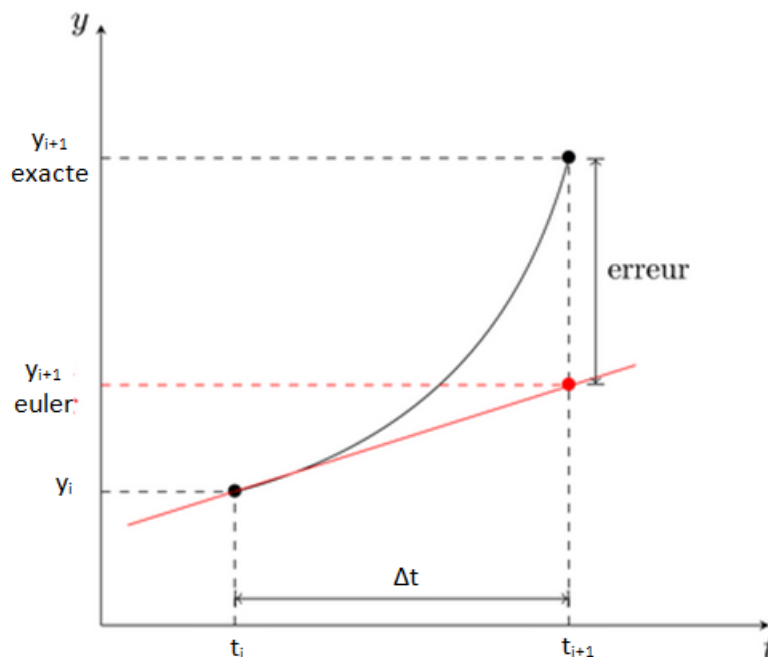


FIGURE 1 – interprétation graphique de la méthode d'Euler

Nous avons aussi appris à résoudre numériquement ce type de problème à l'aide d'un script Python (voir le rappel au paragraphe 1.2 page suivante).

1.2 Script Python 1

Au TP précédent, nous avons vu comment résoudre le problème de Cauchy : $y' + y = 1$ avec $y_0 = 2$.

Nous avons pour cela défini la fonction $F(y, t) = 1 - y$ telle que $y' = F(y, t)$ et une condition initiale $y_0 = 2$.

Puis nous avons défini une fonction $\text{Euler}(G, x_0, a, b, n)$ permettant de résoudre le problème. Cette fonction Euler dépendait des paramètres :

- G : une fonction (ce sera la fonction F définie précédemment) ;
- x_0 : un réel (ce sera la condition initiale y_0) ;
- a, b : les bornes de l'intervalle de temps sur lequel on souhaite résoudre le problème (elles seront définies sous les noms $tmin$ et $tmax$) ;
- n : l'intervalle de temps $(b - a)$ est subdivisé en n sous-intervalles de longueur $\frac{b-a}{n}$.

Après avoir défini les valeurs de $tmin$, $tmax$ et N , il suffisait ensuite d'appeler la fonction $\text{Euler}(F, y_0, tmin, tmax, N)$ puis d'afficher le résultat.

1.3 Script Python 2

Afin de résoudre un problème de Cauchy (par exemple le même que précédemment), il est possible de définir une fonction $\text{Euler}(G, x_0, t)$ qui ne dépend que de 3 paramètres :

- G : une fonction (ce sera la fonction F définie précédemment) ;
- x_0 : un réel (ce sera la condition initiale y_0) ;
- t : un tableau numpy de valeurs du temps comprises entre $tmin$ et $tmax$, avec un pas égal à $\frac{tmax-tmin}{N}$.

Par exemple :

```
T = np.array( [tmin + i * (tmax - tmin) / N for i in range (0, N + 1)] )
```

qui est un tableau contenant $N + 1$ valeurs comprises entre $tmin$ ($i = 0$) et $tmax$ ($i = N$) donc N intervalles de longueur $\frac{tmax-tmin}{N}$.

Ce qui peut être remplacé par :

```
T = np.linspace(tmin, tmax, N+1)
```

Le script devient alors (une fois F , y_0 et T définis) :

```
def Euler(G, x0, t):
    y=x0
    L=[y] # on range la première valeur dans une liste
    for i in range(len(t)-1) :      # attention !!!
        y = y + G(y, t[i]) * (t[i+1]-t[i])    #on calcule les valeurs successives de y
        L.append(y)      # on range la valeur de y dans une liste
    return np.array(L)    # on renvoie un tableau numpy de len(L)=1+(len(t)-1)=len(t) valeurs
#
print(Euler(F, y0, T))
```

Dans la suite, on préférera cette deuxième fonction Euler, en prévision de l'utilisation de la fonction `odeint` prédéfinie dans la bibliothèque `scipy` de Python.

1.4 Applications

- Rappeler l'équation différentielle (en fonction de E et τ) vérifiée par la tension $u_c(t)$ pour un dipôle RC série branché aux bornes d'une source idéale de tension de force électromotrice E .
- Rappeler la solution (dite solution analytique) de cette équation différentielle en fonction de E et τ dans le cas du condensateur initialement déchargé.
- E et τ étant définis, rédiger le script Python permettant d'afficher le graphe de u_c en fonction du temps t (solution analytique) sur l'intervalle $[0, 5\tau]$, à l'aide de la fonction Euler précédente et d'un tableau numpy T adapté (voir paragraphe 1.3).

En TP :

- Résoudre le problème de Cauchy $y' + y = 1$ avec $y_0 = 2$ et tracer la solution numérique.
- Résoudre numériquement l'équation différentielle du condensateur qui se charge, afficher le graphe de la solution numérique ainsi que celui de la solution analytique.

2 La vectorialisation

2.1 Méthode pour résoudre un système d'équations différentielles

Soit l'équilibre chimique suivant : $A + B = C$

On note k_+ la constante cinétique de la réaction dans le sens direct et k_- la constante cinétique de la réaction dans le sens inverse. On peut donc exprimer les vitesses de disparition de A et de B ainsi que la vitesse d'apparition de C et on obtient le système d'équations différentielles suivant :

$$\begin{cases} \frac{d[A]}{dt} = -k_+[A]^p[B]^q + k_-[C]^r \\ \frac{d[B]}{dt} = -k_+[A]^p[B]^q + k_-[C]^r \\ \frac{d[C]}{dt} = k_+[A]^p[B]^q - k_-[C]^r \end{cases}$$

On choisit de vectorialiser le problème en posant $X(t) = \begin{pmatrix} [A](t) \\ [B](t) \\ [C](t) \end{pmatrix}$; donc $\frac{dX}{dt} = \begin{pmatrix} \frac{d[A]}{dt} \\ \frac{d[B]}{dt} \\ \frac{d[C]}{dt} \end{pmatrix}$

Et le système d'équations différentielles peut alors s'écrire sous la forme vectorielle : $\frac{dX}{dt} = F(X, t)$

avec $F\left(X \begin{pmatrix} x \\ y \\ z \end{pmatrix}, t\right) = \begin{pmatrix} -k_+x^py^q + k_-z^r \\ -k_+x^py^q + k_-z^r \\ +k_+x^py^q - k_-z^r \end{pmatrix}$

METHODE _ ÉTAPE 1

On pose $X(t) = \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ \dots \end{pmatrix}$ et on détermine la fonction F telle que $\frac{dX}{dt} = F(X, t)$

De plus, les conditions initiales $\begin{cases} [A](t=0) = [A]_0 \\ [B](t=0) = [B]_0 \\ [C](t=0) = [C]_0 \end{cases}$

peuvent s'écrire vectoriellement : $X(t=0) \begin{pmatrix} [A]_0 \\ [B]_0 \\ [C]_0 \end{pmatrix} = X_0$

METHODE _ ÉTAPE 2

On crée le « vecteur conditions initiales » : $X_0 = \begin{pmatrix} f_1(t=0) \\ f_2(t=0) \\ f_3(t=0) \\ \dots \end{pmatrix}$

En conclusion, le système d'équations différentielles se ramène à un problème de Cauchy vectorialisé à 3 dimensions qu'il faut résoudre numériquement.

METHODE _ ÉTAPE 3

On résout le problème de Cauchy suivant (avec Python; voir paragraphe 3) :

$$\frac{dX}{dt} = F(X, t) \text{ avec } X(t=0) = X_0$$

2.2 Exemple de système d'équations différentielles

Des modèles simples permettent de décrire la dynamique d'une épidémie comme celle causée par le Covid-19. L'un de ces modèles, le modèle *SIR*, est un modèle à « compartiments » : la population est regroupée entre « personnes saines » (susceptibles d'être infectées par le virus) d'effectif S , « personnes infectées » (celles qui portent le virus et sont susceptibles de le transmettre) d'effectif I , et « personnes retirées » (celles qui sont guéries et ne transmettent plus le virus) d'effectif R . La mortalité n'est ici pas prise en compte!

Les effectifs S , I et R évoluent dans le temps et vérifient les équations différentielles suivantes :

$$\begin{cases} \frac{dS(t)}{dt} = -\beta S(t)I(t) \\ \frac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t) \\ \frac{dR(t)}{dt} = \gamma I(t) \end{cases}$$

où β est le taux de transmission et γ le taux de guérison.

Déterminer la fonction F tels que $\frac{dX}{dt} = F(X, t)$.

En TP : compléter le script Python correspondant avec par exemple $S(0) = 1$, $I(0) = 0,5$ et $R(0) = 0$.

2.3 Méthode pour résoudre une équation différentielle du second ordre (et +)

Nous allons maintenant voir que la méthode de vectorialisation permet aussi de ramener une équation différentielle d'ordre supérieur ou égal à 2 à un problème de Cauchy.

Prenons le cas de l'équation différentielle de l'oscillateur harmonique amorti en régime sinusoïdal forcé qui peut s'écrire sous la forme canonique suivante :

$$\begin{cases} \ddot{x}(t) + \frac{\omega_0}{Q}\dot{x}(t) + \omega_0^2 x(t) = F_0 \cos(\omega t) & \iff \ddot{x}(t) = -\frac{\omega_0}{Q}\dot{x}(t) - \omega_0^2 x(t) + F_0 \cos(\omega t) \\ \text{avec les conditions initiales : } x(0) = x_0 \text{ et } \dot{x}(0) = v_0 \end{cases}$$

Astuce :

Si on pose $\dot{\mathbf{x}}_1(t) = \mathbf{x}_2(t)$ alors $\ddot{x}_1(t) = \dot{x}_2(t)$

Cette affirmation est évidente mais elle permet, si on pose $X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$, d'écrire que $\frac{dX}{dt} = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix}$

et donc que $\frac{dX}{dt} = \begin{pmatrix} x_2(t) \\ \ddot{x}_1(t) \end{pmatrix} = \begin{pmatrix} x_2(t) \\ -\frac{\omega_0}{Q}\dot{x}_1(t) - \omega_0^2 x_1(t) + F_0 \cos(\omega t) \end{pmatrix} = \begin{pmatrix} x_2(t) \\ -\frac{\omega_0}{Q}x_2(t) - \omega_0^2 x_1(t) + F_0 \cos(\omega t) \end{pmatrix}$

Conséquence :

Il est alors possible d'introduire une fonction F telle que $\frac{dX}{dt} = F(X, t)$:

$$F : \begin{cases} \mathbb{R}^2 & \longrightarrow \mathbb{R}^2 \\ X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \longmapsto F(X, t) = \begin{pmatrix} x_2 \\ -\frac{\omega_0}{Q}x_2 - \omega_0^2 x_1 + F_0 \cos(\omega t) \end{pmatrix} \end{cases}$$

METHODE

ÉTAPE 0 (dans le cas d'une équation différentielle d'ordre supérieur ou égal à 2)

On pense à l'astuce :

- pour l'ordre $n = 2$, on pose : $\dot{\mathbf{x}}_1 = \mathbf{x}_2$ et donc $\ddot{x}_1 = \dot{x}_2$
- pour l'ordre $n = 3$, on pose : $\dot{x}_1 = x_2$ et $\dot{x}_2 = x_3$; donc $\ddot{x}_1 = \dot{x}_3$
- etc.

ÉTAPE 1

On pose $X(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \dots \end{pmatrix}$ et on détermine la fonction F de $\mathbb{R}^n$ dans $\mathbb{R}^n$ telle que $\frac{dX}{dt} = F(X, t)$

Et si, à $t = 0$, $x(t = 0) = x_0$ et $\dot{x}(t = 0) = v_0$, on crée le « vecteur conditions initiales » : $X_0 = \begin{pmatrix} x_0 \\ v_0 \end{pmatrix}$

ÉTAPE 2

On crée le « vecteur conditions initiales » : $X_0 = \begin{pmatrix} x_1(t = 0) \\ x_2(t = 0) \\ x_3(t = 0) \\ \dots \end{pmatrix}$

ÉTAPE 3

On résout le problème de Cauchy suivant (avec Python ; voir paragraphe 3) :

$$\frac{dX}{dt} = F(X, t) \text{ avec } X(t = 0) = X_0$$

2.4 Exemple d'équation différentielle d'ordre 2

Soit un circuit RLC série et soit $q(t)$ la charge du condensateur à une date t . On considérera que le condensateur possédait initialement la charge Q_0 et qu'aucun courant ne circulait dans le circuit.

L'équation différentielle vérifiée par la fonction $q(t)$ est :

$$\ddot{q}(t) + \frac{R}{L}\dot{q}(t) + \frac{1}{LC}q(t) = 0$$

Déterminer la fonction F tels que $\frac{dX}{dt} = F(X, t)$.

Déterminer aussi le « vecteur conditions initiales » X_0 dans le cas où $Q_0 = 1$ C par exemple.

En TP : compléter le script Python correspondant en affichant 3 périodes.

3 Etudes de différents oscillateurs

3.1 Oscillateur non amorti : résolution avec la fonction Euler

L'objectif est de résoudre l'équation différentielle de l'oscillateur mécanique harmonique vertical :

$$z''(t) + \omega_0^2 z(t) = 0$$

avec $z(0) = 1,0 \text{ cm}$, $z'(0) = 0 \text{ cm} \cdot \text{s}^{-1}$ et $\omega_0 = 0,50 \text{ rad} \cdot \text{s}^{-1}$.

Vectorialiser le problème puis compléter le script ci-dessous (on choisira X_0 de façon cohérente avec la courbe ci-dessous).

```
import numpy as np
#
def F(X,t):
    return .....
#
X0=..... # conditions initiales
```

Déterminer la solution analytique puis compléter le script ci-dessous permettant de la tracer sur 3 périodes.

```
import numpy as np
import matplotlib.pyplot as plt
#
omega0=0.5
tmin,tmax,N=0,.....,200
T=np.linspace(tmin,tmax,N+1)
#
plt.plot(.....,'r',label='solution analytique')
```

En TP : dans Capytale, compléter le script Python afin de tracer la solution analytique ainsi que la solution numérique.

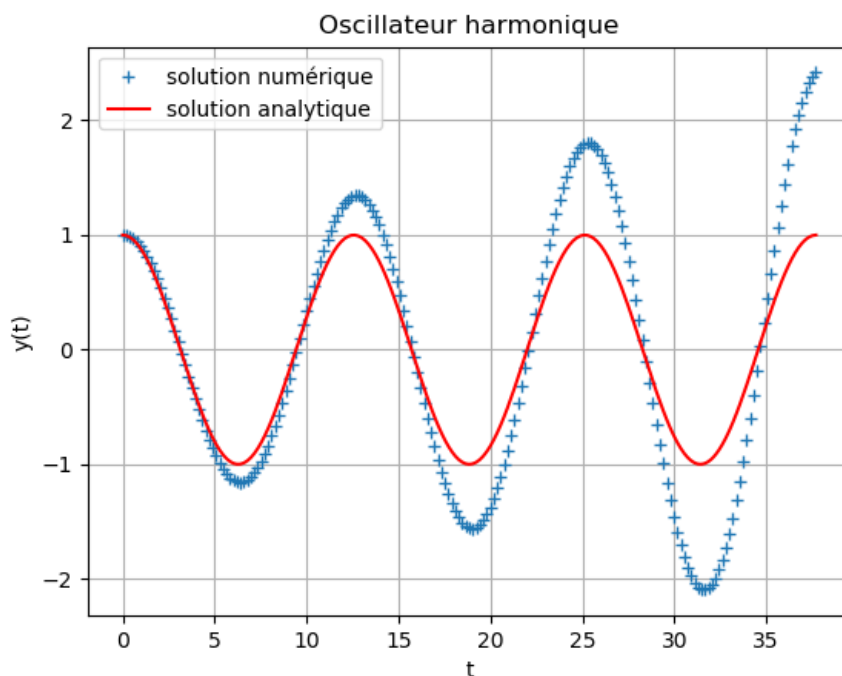


FIGURE 2 – Résolution de l'équation différentielle de l'oscillateur harmonique avec la méthode d'Euler

Commenter.

.....

.....

3.2 Oscillateur non amorti : résolution avec la fonction odeint

Parmi les méthodes numériques de résolution des équations différentielles, la fonction `odeint` est déjà prédéfinie dans python; elle figure au programme. Nous ne nous préoccupons pas de la méthode mathématique sous-jacente, nous nous contenterons de l'utiliser.

Sa syntaxe est identique à la fonction `Euler` introduite précédemment (fonction des 3 paramètres F , $X0$ et T). Pour l'utiliser, il suffit de remplacer `Euler` par `odeint` dans le programme précédent, sous réserve de l'avoir importée avec la commande : `from scipy.integrate import odeint`.

Dans Capytale, compléter le script Python correspondant puis commenter.

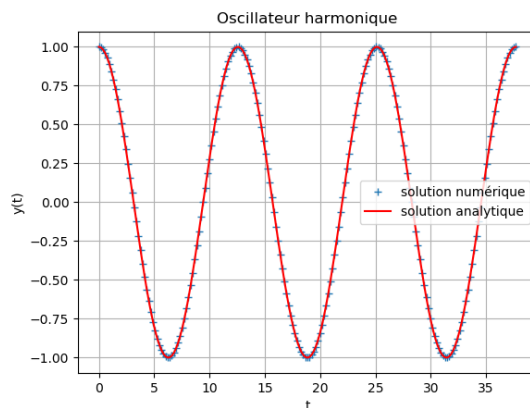


FIGURE 3 – Résolution de l'équation différentielle de l'oscillateur harmonique avec `odeint`

3.3 Cas du pendule : influence des conditions initiales

Dans le cas d'un pendule simple, de longueur L , l'équation différentielle non linéaire vérifiée par l'angle θ entre la verticale et le fil du pendule est :

$$\theta''(t) + \omega_0^2 \sin(\theta(t)) = 0 \quad \text{avec} \quad \omega_0 = \sqrt{\frac{g}{L}}$$

Nous savons que, dans le cas des petits angles, $\sin(\theta) \approx \theta$ et l'équation différentielle devient donc celle de l'oscillateur harmonique de pulsation $\omega_0 = \sqrt{\frac{g}{L}}$ donc de période $T_0 = 2\pi\sqrt{\frac{L}{g}}$ indépendante de l'amplitude initiale (et de la vitesse initiale; donc de manière générale, des conditions initiales).

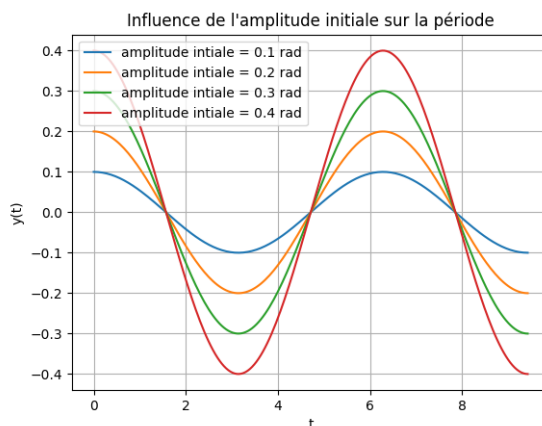


FIGURE 4 – Isochronisme des petites oscillations

Dans le cas de l'équation différentielle non linéaire $\theta''(t) + \omega_0^2 \sin(\theta(t)) = 0$, compléter le script Python permettant de tracer le graphe ci-dessous.

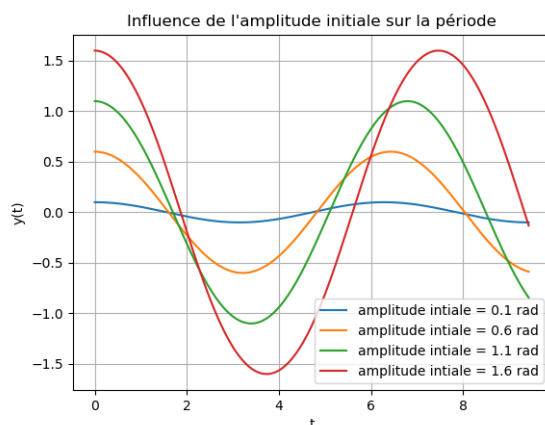


FIGURE 5 – influence de l'amplitude initiale sur la période des oscillations

Aide :

- il faudra créer une liste L de valeurs de y_0 puis tracer sur le même graphe les solutions numériques de l'équation différentielle non linéaire.
- pour la légende, on pourra écrire : `label='amplitude initiale = '+str(y0)+' rad'`.

Commenter.

3.4 Oscillateur amorti

Résoudre l'équation différentielle du paragraphe 2.4. avec la fonction `odeint` et avec les valeurs numériques (peu réalistes ; nous en discuterons...) suivantes : $L = 9 \text{ H}$; $C = 1 \text{ F}$; $Q_0 = 1 \text{ C}$.

Pour R , on optera pour $R = a * R_c = a * \frac{1}{2} \sqrt{\frac{L}{C}}$ avec $a = 0,7$ dans un premier temps puis vous pourrez tester d'autres valeurs de a , supérieures ou inférieures à 1.

Compléter le script Python de manière à tracer l'évolution de la charge q en fonction du temps sur $5T_0$ ainsi que l'évolution de l'intensité i en fonction du temps.

Aide : Afficher le tableau X et s'interroger sur ce que représentent ses deux colonnes.

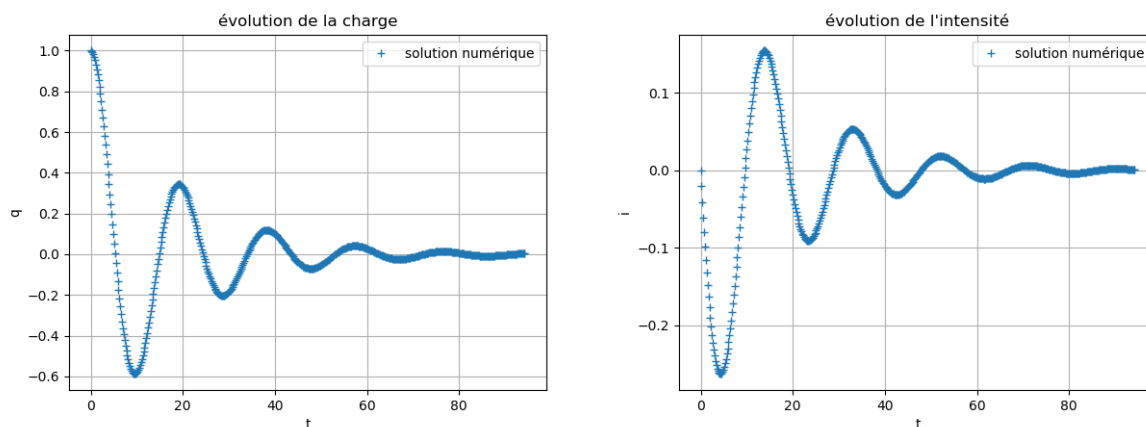


FIGURE 6 – Tracés dans le cas $R = 0,7 \times R_c$

3.5 Portrait de phase

Compléter le script Python afin de tracer le portrait de phase de l'oscillateur précédent (circuit RLC série) c'est-à-dire le graphe obtenu en plaçant q en abscisses et $\frac{dq}{dt}$ en ordonnées.

Tester le cas $R = 0,7 \times Rc$ puis $R = 0 \Omega$.

Commenter.

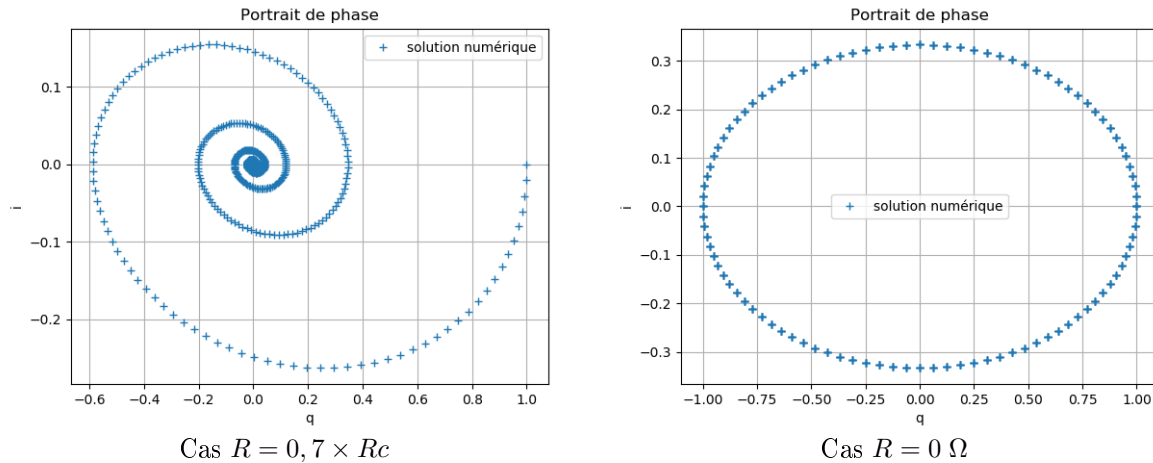


FIGURE 7 – Portraits de phase

4 Introduction au régime sinusoïdal forcé

Résoudre l'équation différentielle ci-dessous, sur 5 périodes propres, en reprenant les valeurs précédentes pour R , L , C et pour les conditions initiales. Choisir aussi $\omega = 0,5 \text{ rad} \cdot \text{s}^{-1}$ et $E = 0,5 \text{ V}$. Compléter le script Python (cellule 6).

$$\ddot{q}(t) + \frac{R}{L}\dot{q}(t) + \frac{1}{LC}q(t) = CE\cos(\omega t)$$

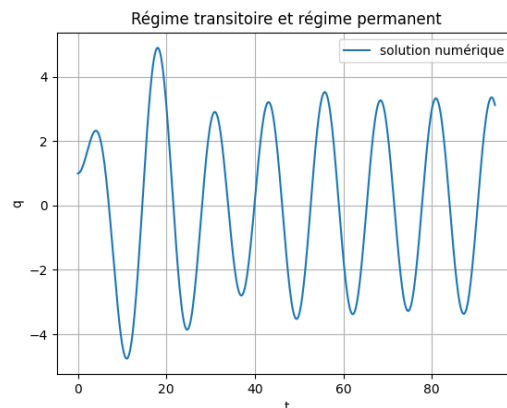


FIGURE 8 – Cas d'un second membre sinusoïdal

- Mesurer à l'aide du curseur la période des oscillations en régime permanent. Comparer à la période propre T_0 et à la période T du GBF. Conclure
- Mesurer approximativement la durée du régime transitoire et comparer à la constante de temps $\tau = \frac{L}{R}$. Conclure.