# M2_Euler_p2_corr

Préambule python :

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt, cos, sin, exp, pi
from scipy.integrate import odeint
```

Modélisation 2 : Méthode d' Euler - Partie II

## 1 Problème : le pendule

**Q 1 :** Compléter :

- équation différentielle pour un pendule idéal *sans approximation des petits angles* :

$$\ddot{\theta} + \omega_0^2 sin\theta = 0$$

- avec l'approximation des petits angles :

$$\ddot{\theta} + \omega_0^2 \theta = 0$$

- solution, avec l'approximation, pour l'angle initial $\theta_i$ et une vitesse initiale nulle :

$$\theta(t) = \theta_i cos(\omega_0 t)$$

Dans la suite, nous allons résoudre numériquement l'équation du pendule sans approximation des petits angles, pour la valeur $\omega_0 = 1\ rad.s^{-1}$. Pour cela, on ré-écrit l'équation différentielle du second ordre en deux équations du premier ordre :

$$\begin{cases} \dot{\theta}(t) = \omega(t) \\ \dot{\omega}(t) = -sin(\theta(t)) \end{cases}$$

Ou, sous forme vectorielle $\dot{Y} = F(Y,t)$ avec :

$$Y(t) = \begin{pmatrix} \theta(t) \\ \omega(t) \end{pmatrix} \text{ et } F(Y,t) = \begin{pmatrix} \omega(t) \\ -sin(\theta(t)) \end{pmatrix}$$

Le but de la suite de cette partie est de résoudre numériquement, avec une méthode d'Euler, ce problème entre les temps $t_i$ et $t_f$, en subdivisant cet intervalle en $n$ intervalles égaux. Pour cela , nous allons créer trois listes ou tableaux, de longueur $n+1$ :

- `T` permettra de stocker les valeurs du temps $t_0 = t_i, t_1, t_2, ..., t_{n-1}, t_n = t_f$
- `Theta` permettra de stocker les valeurs de l'angle $\theta(t)$, au cours du temps $\theta_i = \theta(t_0)$, $\theta_1 = \theta(t_1)$, $\theta_2 = \theta(t_2)$, ..., $\theta_n = \theta(t_n)$
- `Omega` permettra de stocker les valeurs de la pulsation $\omega(t)$, au cours du temps $\omega_0 = \omega(t_0)$, $\omega_1 = \omega(t_1)$, $\omega_2 = \omega(t_2)$, ..., $\omega_n = \omega(t_n)$

**Q 2 :** Ecrire la fonction `Pendule_Euler(theta_i,omega_i,ti,tf,n)` retournant les listes ou tableaux `T`, `Theta` et `Omega`, grâce à la méthode d'Euler, appliquée entre *ti* et *tf*, avec pour conditions initiales `theta_i` $= \theta(t_i)$ et `omega_i` $= \omega(t_i) = \dot{\theta}(t_i)$ .

Première version :

[22]:
```
def Pendule_Euler(theta_i,omega_i,ti,tf,n):
    # pas de la résolution :
    h = (tf-ti)/n
    # Création des tableaux de 0 pour les 3 variables t, theta et omega
    T = np.zeros(n+1)
    Theta = np.zeros(n+1)
    Omega = np.zeros(n+1)
    # Conditions initiales
    T[0] = ti
    Theta[0] = theta_i
    Omega[0] = omega_i
    # Boucle de la méthode d'Euler
    for j in range(1,n+1):
        T[j] = T[j-1] + h
        Theta[j] = Theta[j-1]+h*Omega[j-1]
        Omega[j] = Omega[j-1]-h*sin(Theta[j-1])
    return T,Theta,Omega
```

Autres versions :

[23]:
```
def Pendule_Euler(theta_i,omega_i,ti,tf,n):
    # pas de la résolution :
    h = (tf-ti)/n
    # Conditions initiales
    T = [ti]
    Theta = [theta_i]
    Omega = [omega_i]
    # Boucle de la méthode d'Euler
    for j in range(1,n+1):
        T.append(T[j-1]+h)
        Theta.append(Theta[j-1]+h*Omega[j-1])
        Omega.append(Omega[j-1]-sin(Theta[j-1])*h)
    return T,Theta,Omega
```

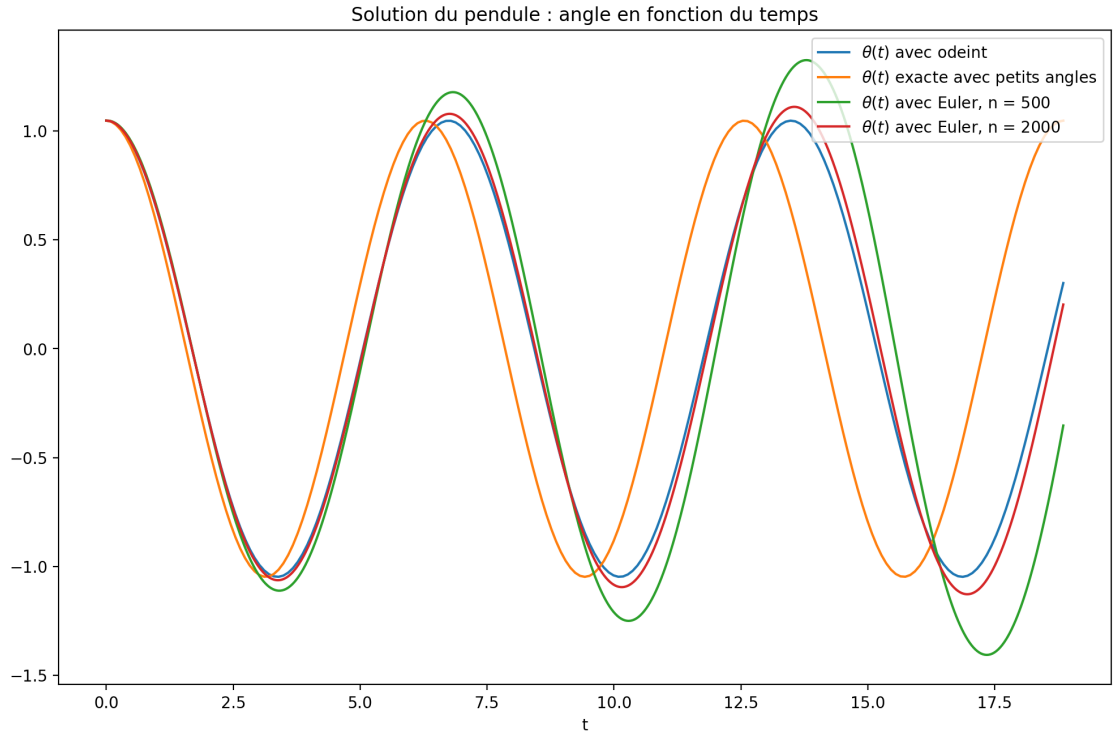
[25]:
```
def Pendule_Euler(theta_i,omega_i,ti,tf,n):
    # pas de la résolution :
```

```
    h = (tf-ti)/n
    # Conditons initiales
    T = [ti]
    Theta = [theta_i]
    Omega = [omega_i]
    # Pour l'instant j = 0
    tj = ti
    theta_j = theta_i
    omega_j = omega_i
    for k in range(n):
        tj = tj + h # Nouvelle valeur de t
        # Nouvelles valeurs de theta et omega
        # Attention ici, il faut mettre  à jour ces valeurs "en même temps",␣
 ↪savez-vous pourquoi ?
        theta_j, omega_j = theta_j+h*omega_j, omega_j-h*sin(theta_j)
        T.append(tj)
        Theta.append(theta_j)
        Omega.append(omega_j)
    return T,Theta,Omega
```

**Q 3 :** Comparer avec la résolution numérique fournie avec la bibliothèque `scipy` (fonction `odeint`) et la solution exacte avec approximation des petits angles, pour les conditions initiales $\theta_i = \frac{\pi}{3}$ et une vitesse initiale nulle. Pour cela, on complétera le code du fichier "M2_Euler.py" :

[23]:
```
def pend(y, t): # fonction définissant les deux équations différentielles␣
 ↪couplées
    theta, omega = y
    dydt = [omega, -np.sin(theta)]
    return dydt

# conditions initiales :
theta_i=np.pi/3 # C.I. pour l'angle theta
omega_i=0 # C.I. pour oméga, c'est-à-dire la vitesse angulaire.
yi = [theta_i, omega_i]

ti=0 # Début de l'intervalle de temps
tf=6*pi # fin de l'intervalle de temps
T=np.linspace(ti,tf,201) # liste des points de l'intervalle temporel de␣
 ↪résolution

ratio = 1.5 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
# Graphe avec la fonction "toute faite" odeint
plt.plot(T, odeint(pend,yi,T)[:,0],label=r'$ \theta (t)$ avec odeint')

# Graphe de la solution exacte avec approx des petits angles
plt.plot(T, theta_i*np.cos(T),label=r'$ \theta (t)$ exacte avec petits angles')
```

```
#Graphe avec Pendule_Euler :
T, Theta, Omega = Pendule_Euler(theta_i,omega_i,ti,tf,500)
plt.plot(T,Theta,label=r'$ \theta (t)$ avec Euler, n = 500')
T, Theta, Omega = Pendule_Euler(theta_i,omega_i,ti,tf,2000)
plt.plot(T, Theta,label=r'$ \theta (t)$ avec Euler, n = 2000')

plt.xlabel("t")
plt.title("Solution du pendule : angle en fonction du temps")
plt.legend()
plt.show()
```

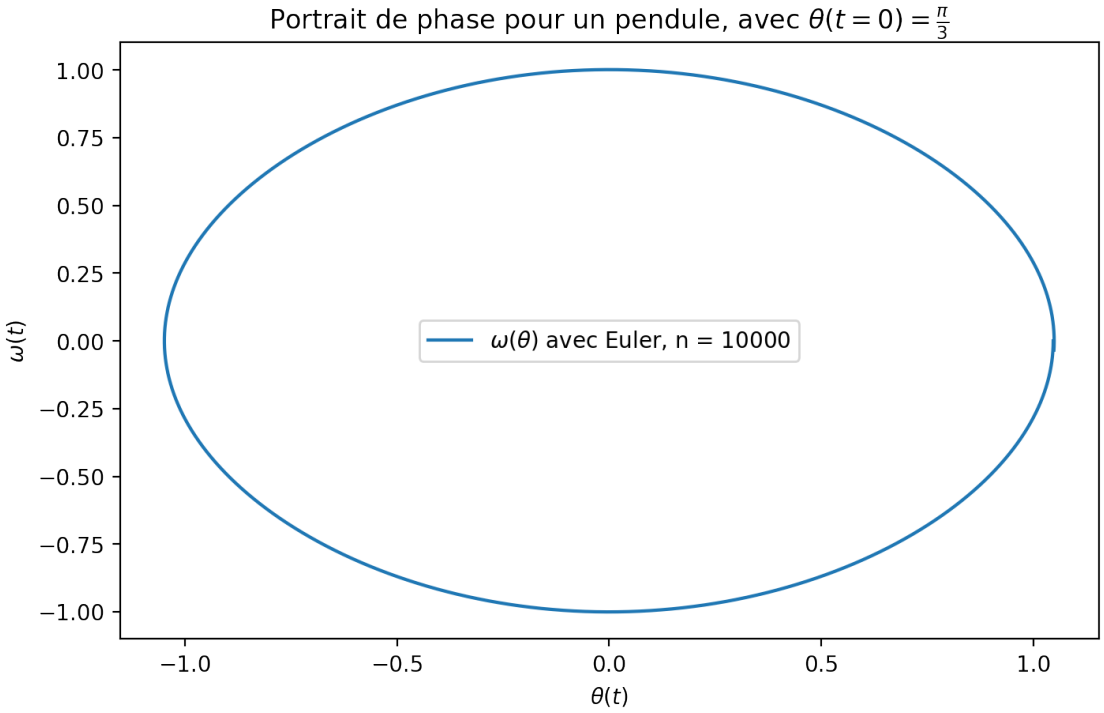



Dans la suite, on va réaliser le portrait de phase du système pendule : cela consiste à tracer un graphique de la vitesse angulaire en fonction de l'angle. Ainsi, on placera en abscisse l'angle $\theta$ et en ordonnée la vitesse angulaire $\omega$, et on obtient la trajectoire dans l'espace des phases en faisant varier le temps.

**Q 4 :** Écrire un script permettant de réaliser le portrait de phase du système pendule, à partir de `ti=0`, pour les conditions initiales $\theta_0 = \frac{\pi}{3}$ et une vitesse initiale nulle. On utilisera uniquement la fonction `Pendule_Euler`. Quel temps final `tf` faut-il considérer ?

[27]:
```
ti=0 # Début de l'intervalle de temps
tf=2*pi+0.5 # fin de l'intervalle de temps : 2 pi est approximativement une␣
↪période (inutile de faire plus, et si on met moins, on aura pas toute la␣
↪trajectoire !)
```

```
theta_i=np.pi/3 # C.I. pour l'angle theta
omega_i=0 # C.I. pour oméga, c'est-à-dire la vitesse angulaire.

# Graphe
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
T, Theta, Omega = Pendule_Euler(theta_i,omega_i,ti,tf,10000)
plt.plot(Theta, Omega,label=r'$ \omega (\theta)$ avec Euler, n = 10000')
plt.legend()
plt.xlabel(r'$\theta (t)$')
plt.ylabel(r'$\omega (t)$')
plt.title(r"Portrait de phase pour un pendule, avec $ \theta␣
 ↪(t=0)=\frac{\pi}{3} $")
plt.show()
```



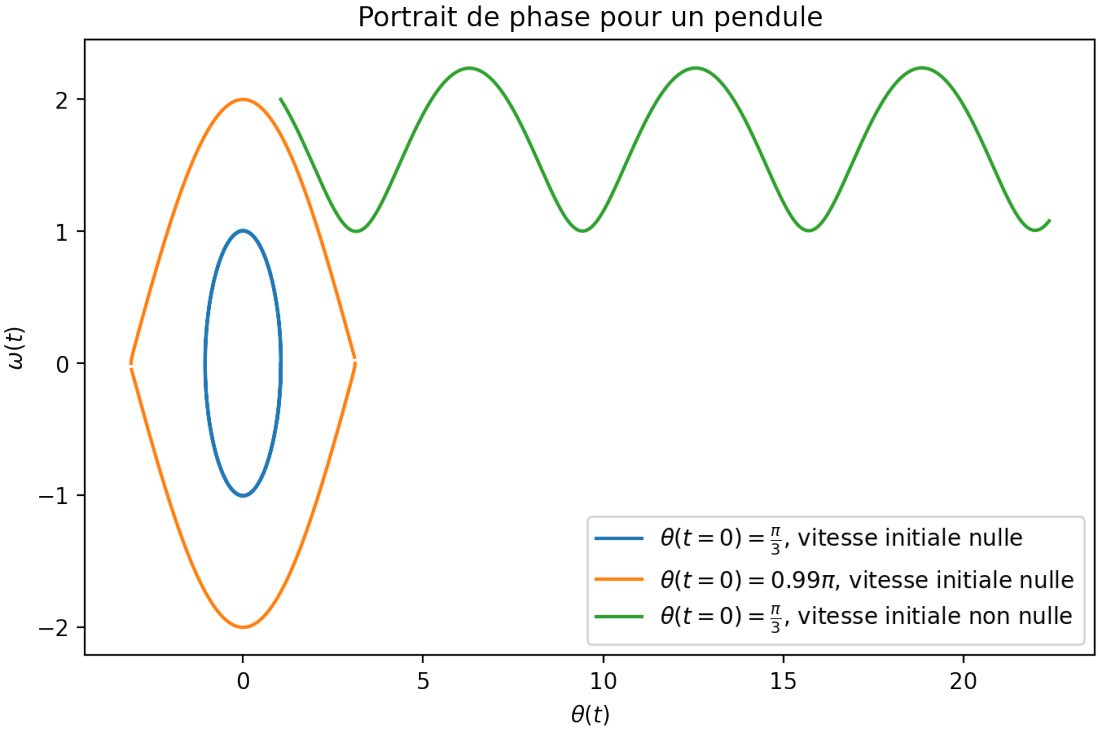

Pour l'instant, les angles restent raisonnables, et la trajectoire dans l'espace des phases ressemble à une ellipse (c'est exactement une ellipse avec l'approximation des petits angles).

Si on se rapproche de $\pi$ et $-\pi$, on perçoit une forme différentes, surtout au niveau des extrema d'angles, et la trajectoire "prend" beaucoup plus de temps pour "boucler" (effectivement, aux alentours de $\theta = \pi$ et $\theta = -\pi$, le mouvement du pendule devent infiniement lent !).

Enfin, si on aide le pendule en lui donnant une vitesse initiale suffisante, il peut tourner sans fin !

[28]:
```
ti=0 # Début de l'intervalle de temps
tf=14 # fin de l'intervalle de temps
```

```
# Graphe
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.plot(Pendule_Euler(np.pi/3,0,ti,tf,10000)[1],Pendule_Euler(np.pi/
 ↪3,0,ti,tf,10000)[2],label=r'$ \theta (t=0)=\frac{\pi}{3} $, vitesse initiale␣
 ↪nulle')
plt.plot(Pendule_Euler(+0.99*np.pi,0,ti,10,50000)[1],Pendule_Euler(0.99*np.
 ↪pi,0,ti,10,50000)[2],label=r'$ \theta (t=0)=0.99\pi $, vitesse initiale␣
 ↪nulle', color='C1')
plt.plot(Pendule_Euler(-0.99*np.pi,0,ti,10,50000)[1],Pendule_Euler(-0.99*np.
 ↪pi,0,ti,10,50000)[2], color='C1')
plt.plot(Pendule_Euler(np.pi/3,2,ti,tf,10000)[1],Pendule_Euler(np.pi/
 ↪3,2,ti,tf,10000)[2],label=r'$ \theta (t=0)=\frac{\pi}{3} $, vitesse initiale␣
 ↪non nulle',color='C2')

plt.legend()
plt.xlabel(r'$\theta (t)$')
plt.ylabel(r'$\omega (t)$')
plt.title(r"Portrait de phase pour un pendule")
plt.show()
```




**Q 5** : Interpréter la forme de la trajectoire observée, et donner le sens de parcours.

La trajectoire forme une courbe fermée (elle "boucle" sur elle-même) : logique car le mouvement est périodique ! Elle se parcourt dans le sens horaire si les angles sont orientés dans le sens direct.

**Question 6 :** Ecrire la nouvelle fonction `Pendule_Euler_f(theta_i,omega_i,ti,tf,f,n)` re-

tournant les listes `T`, `Theta` et `Omega`, grâce à la méthode d'Euler, appliquée entre *ti* et *tf*, avec pour conditions initiales `theta_i` $= \theta(t_i)$ et `omega_i` $= \omega(t_i) = \dot{\theta}(t_i)$, en prenant cette fois-ci en compte des frottements fluides (force de frottement de module $F = mfv$).

L'équation différentielle s'écrit alors

$$\ddot{\theta} + f\dot{\theta} + \omega_0 sin\theta = 0$$

et le système :

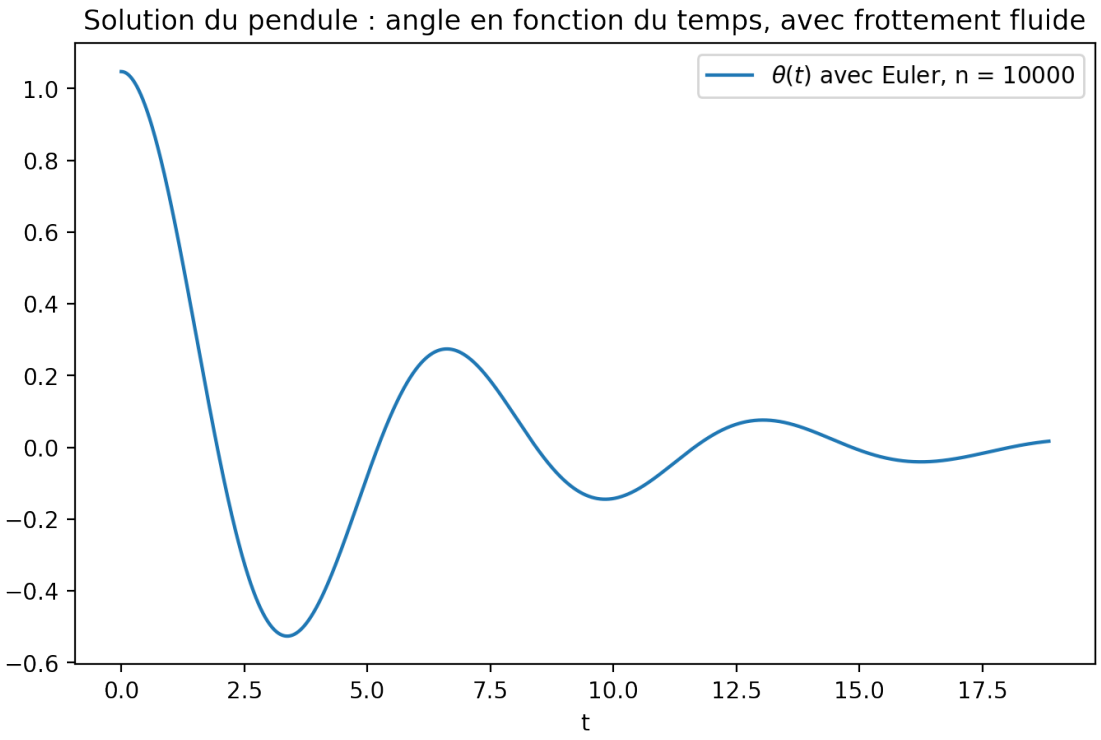
$$\begin{cases} \dot{\theta}(t) = \omega(t) \\ \dot{\omega}(t) = -f\omega(t) - sin(\theta(t)) \end{cases}$$

Et voici la fonction :

[33]:
```
def Pendule_Euler_f(theta_i,omega_i,ti,tf,f,n):
    # pas de la résolution :
    h = (tf-ti)/n
    T=[ti]
    Theta=[theta_i]
    Omega=[omega_i]
    for j in range(1,n+1):
        T.append(T[j-1]+h)
        Theta.append(Theta[j-1]+h*Omega[j-1])
        Omega.append(Omega[j-1]-(f*Omega[j-1]+sin(Theta[j-1]))*h)
    return T,Theta,Omega
```

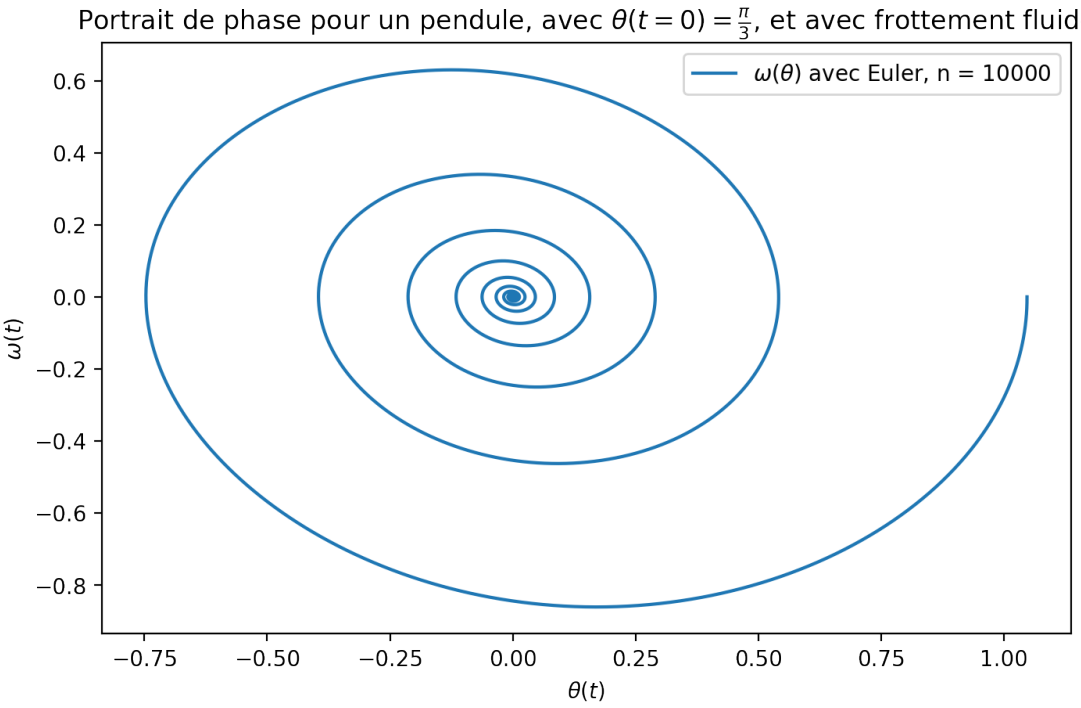
Vérification du "bon fonctionnement" :

[34]:
```
ti=0 # Début de l'intervalle de temps
tf=6*pi # fin de l'intervalle de temps

theta_i=np.pi/3 # C.I. pour l'angle theta
omega_i=0 # C.I. pour oméga, c'est-à-dire la vitesse angulaire.

ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)

T, Theta, Omega = Pendule_Euler_f(theta_i,omega_i,ti,tf,0.4,10000)
plt.plot(T, Theta,label=r'$ \theta (t)$ avec Euler, n = 10000')

plt.xlabel("t")
plt.title("Solution du pendule : angle en fonction du temps, avec frottement␣
 ↪fluide")
plt.legend()
plt.show()
```




**Q 20 :** Tracer le nouveau portrait de phase, pour les mêmes conditions que dans la question 3, pour $f = 0,2\ s^{-1}$. Qu'observe-t-on ? Quel temps final `tf` faut-il considérer ?

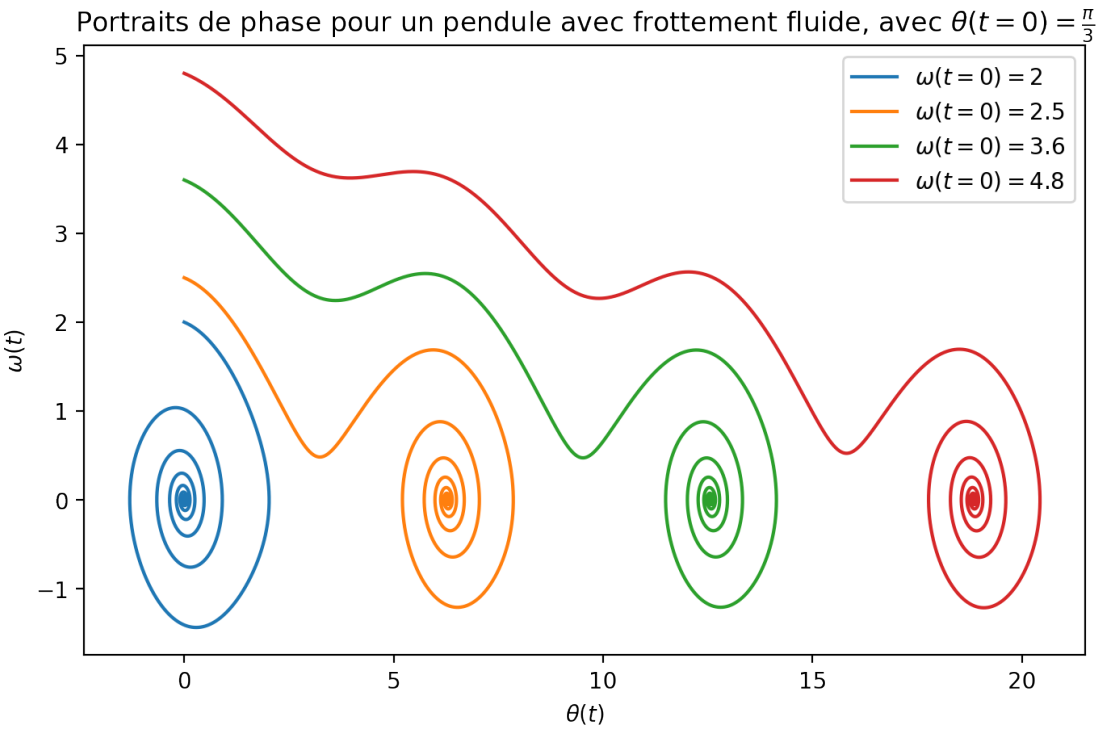
[35]:
```
ti=0 # Début de l'intervalle de temps
tf=20*pi # fin de l'intervalle de temps : le mouvement n'est plus périodique,␣
 ↪il est plus interessant ici d'aller bien plus loin que la pseudo-période 2pi

theta_i=np.pi/3 # C.I. pour l'angle theta
omega_i=0 # C.I. pour oméga, c'est-à-dire la vitesse angulaire.

T, Theta, Omega = Pendule_Euler_f(theta_i,omega_i,ti,tf,0.2,10000)

# Graphe
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.plot(Theta,Omega,label=r'$ \omega (\theta)$ avec Euler, n = 10000')
plt.legend()
plt.xlabel(r'$\theta (t)$')
plt.ylabel(r'$\omega (t)$')
plt.title(r"Portrait de phase pour un pendule, avec $ \theta␣
 ↪(t=0)=\frac{\pi}{3} $, et avec frottement fluide")
plt.show()
```




Le point de coordonnées (0,0) est nommé *attracteur*, car il semble attirer la trajectoire.

**Q 7** : En modifiant la vitesse initiale, montrer qu'il existe plusieurs attracteurs.

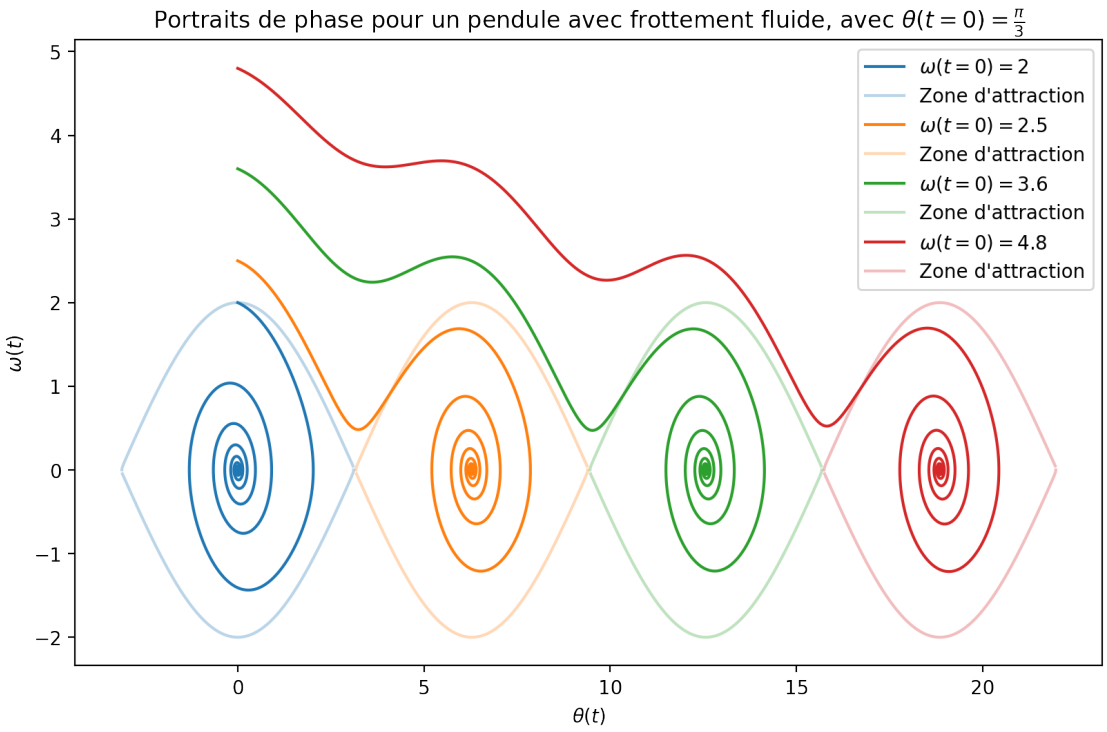
[37]:
```
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)

ti=0 # Début de l'intervalle de temps
tf=20*pi # fin de l'intervalle de temps : le mouvement n'est plus périodique,␣
 ↪il est plus interessant ici d'aller bien plus loin que la pseudo-période 2pi

for omega_i in [2,2.5,3.6,4.8]:
    # conditions initiales :
    theta_i=0
    # résolution
    T, Theta, Omega = Pendule_Euler_f(theta_i,omega_i,ti,tf,0.2,10000)
    #Graphes
    plt.plot(Theta,Omega,label="$\omega (t=0) = $"+str(round(omega_i,2)))

plt.legend()
plt.xlabel(r'$\theta (t)$')
plt.ylabel(r'$\omega (t)$')
plt.title(r"Portraits de phase pour un pendule avec frottement fluide, avec␣
 ↪$\theta (t=0)=\frac{\pi}{3}$")
plt.show()
```




Plus on donne une vitesse initiale importante au pendule, plus il va effectuer de tours complets, avec d'être attiré par un attracteur : en effet, au fur et à mesure, il perd en énergie à cause des frottements. Les attracteurs sont les points de coordonnées $(\theta = 2n\pi, \omega = 0), n \in \mathbb{Z}$; soit les points de vitesse nulle et avec la masse en bas : la masse est donc au repos à la position d'équilibre stable. Une fois la trajectoire "entrée" dans la zone d'attraction d'un attracteur, elle va converger vers celui-ci. Avec ces zones :

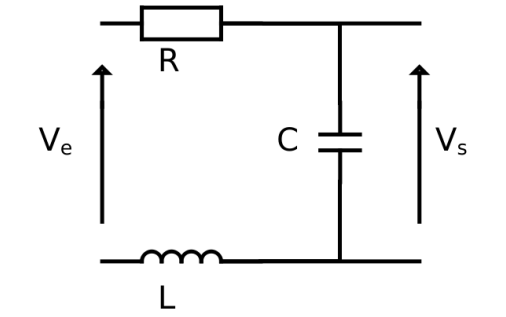
[39]:
```
ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
ti=0 # Début de l'intervalle de temps
tf=20*pi # fin de l'intervalle de temps : le mouvement n'est plus périodique,␣
 ↪il est plus interessant ici d'aller bien plus loin que la pseudo-période 2pi

k=0

for omega_i in [2,2.5,3.6,4.8]:
    # conditions initiales :
    theta_i=0
    # résolutions
    T, Theta, Omega = Pendule_Euler_f(theta_i,omega_i,ti,tf,0.2,10000)
    T, Theta2, Omega2 = Pendule_Euler(2*k*np.pi+0.99*np.pi,0,ti,10,50000)
    T, Theta3, Omega3 = Pendule_Euler(2*k*np.pi-0.99*np.pi,0,ti,10,50000)
    #Graphes
    plt.plot(Theta,Omega,label="$\omega (t=0) = $"+str(round(omega_i,2)),␣
 ↪color='C'+str(k))
```

```
    plt.plot(Theta2,Omega2,label=r"Zone d'attraction", color='C'+str(k),alpha=0.
↪3)
    plt.plot(Theta3,Omega3, color='C'+str(k),alpha=0.3)
    k+=1

plt.legend()
plt.xlabel(r'$\theta (t)$')
plt.ylabel(r'$\omega (t)$')
plt.title(r"Portraits de phase pour un pendule avec frottement fluide, avec␣
↪$\theta (t=0)=\frac{\pi}{3}$")
plt.show()
```




## 2 Problème : Régime transitoire pour un circuit RLC

On considère un circuit RLC série alimenté, à partir du temps $t = 0$, par un GBF à la pulsation $\omega$ : pour $t > 0$, $v_e(t) = cos(\omega t)$. La sortie de ce filtre étudiée est prise aux bornes du condensateur.




**Q 8 :** Ecrire la fonction permettant de résoudre numériquement, via une méthode d'Euler, l'équation différentielle sur la tension aux bornes du condensateur.

*Conseil : Utiliser les variables $\omega_0$ et $Q$ plutôt que $R$, $L$ et $C$.*

Pour un tel circuit, la fonction de transfert s'écrit :

$$H(\omega) = \frac{1}{1 + j\frac{x}{Q} - x^2}$$

avec $Q = \frac{1}{R}\sqrt{\frac{L}{L}}$ et $x = \frac{\omega}{\omega_0} = \omega\sqrt{LC}$

L'équation différentielle liée au système s'écrit :

$$v_e(t) = LC\frac{d^2 v_s}{dt^2} + RC\frac{dv_s}{dt} + v_s(t)$$

Soit :

$$\omega_0^2 v_e(t) = \frac{d^2 v_s}{dt^2} + \frac{\omega_0}{Q}\frac{dv_s}{dt} + \omega_0^2 v_s(t)$$

Ou, sous forme de système différentiel avec seulement des équations d'ordre 1 :

$$\begin{cases} \dot{v_s}(t) = u(t) \\ \dot{u}(t) = -\frac{\omega_0}{Q}u(t) + \omega_0^2(v_e(t) - v_s(t)) \end{cases}$$

Il reste alors à écrire la fonction pour résoudre numériquement, par la méthode d'Euler, cette équation :

[40]:
```
def RLC_Euler_f(v_i,u_i,ti,tf,omega0,Q,omega,n):
    # pas de la résolution :
    h=(tf-ti)/n
    T, U, Vs, Ve = np.zeros(n+1), np.zeros(n+1), np.zeros(n+1), np.zeros(n+1)
    T[0] = ti
    U[0] = u_i
    Vs[0] = v_i
    for j in range(1,n+1):
        T[j] = T[j-1]+h
```

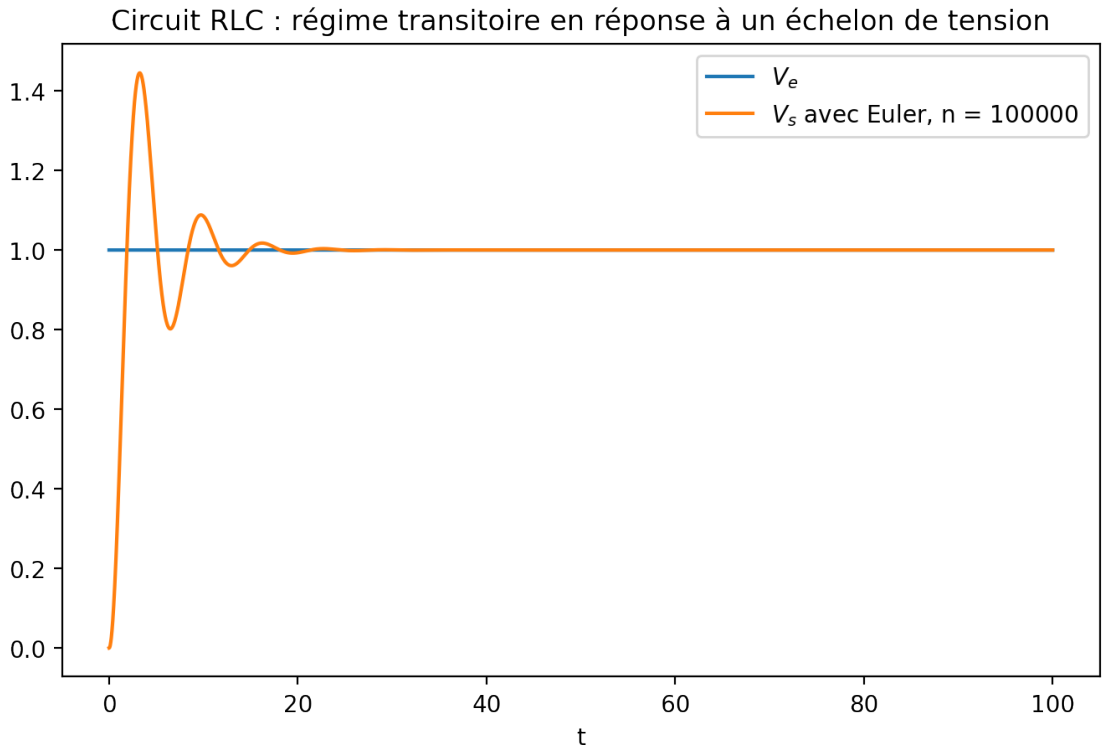
```
        Ve[j] = cos(omega*T[j])
        Vs[j] = Vs[j-1]+h*U[j-1]
        U[j] = (1-h*omega0/Q)*U[j-1]+(Ve[j-1]-Vs[j-1])*h*omega0**2
    return T,Ve,Vs,U

# Seconde version, avec des listes :
def RLC_Euler_f(v_i,u_i,ti,tf,omega0,Q,omega,n):
    # pas de la résolution :
    h=(tf-ti)/n
    T=[ti]
    U=[u_i]
    Vs=[v_i]
    Ve=[cos(omega*ti)]
    for j in range(1,n+1):
        T.append(T[j-1]+h)
        Ve.append(cos(omega*T[j]))
        Vs.append(Vs[j-1]+h*U[j-1])
        U.append((1-h*omega0/Q)*U[j-1]+(Ve[j-1]-Vs[j-1])*h*omega0**2)
    return T,Ve,Vs,U
```

**Q 9 :** Régime transitoire en réponse à un échelon de tension : tracer le régime transitoire observé aux bornes du condensateur, en réponse à un échelon de tension. Observer les différents régimes possibles, et tracer pour chacun le portrait de phase.

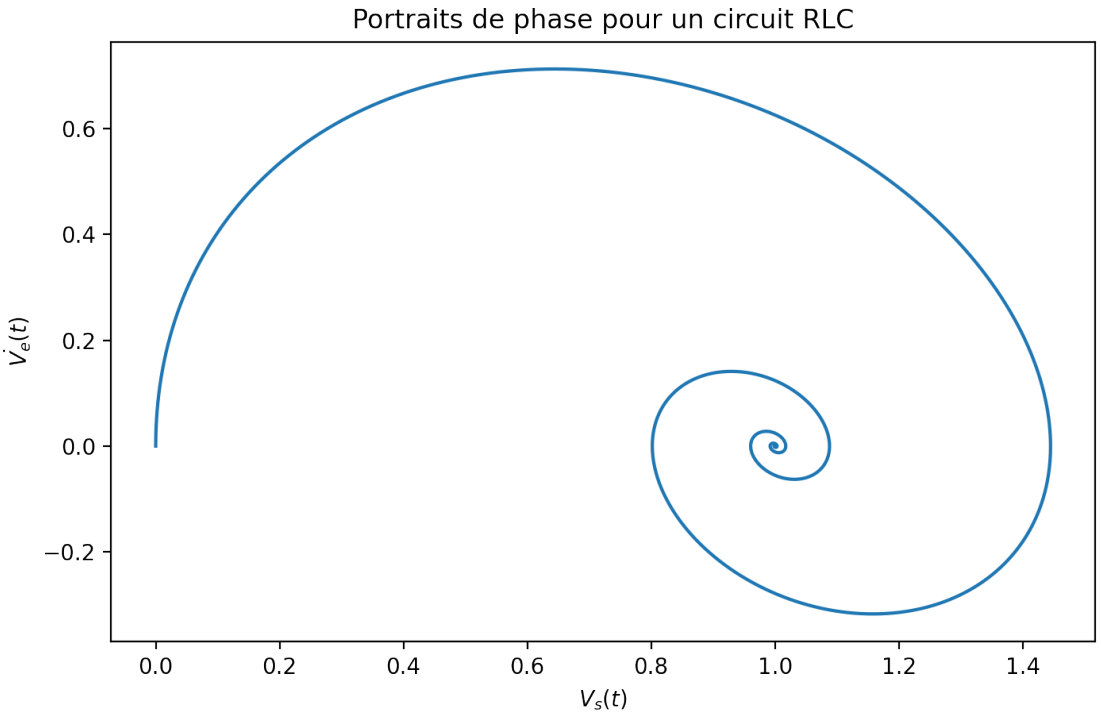
*Aide : la fonction $v_e(t)$ représente une tension continue si on prend $\omega = 0$ !*

[41]:
```
T,Ve,Vs,U = RLC_Euler_f(0,0,0,100,1,2,0,100000)
```

[42]:
```
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$ avec Euler, n = 100000')


plt.xlabel("t")
plt.title("Circuit RLC : régime transitoire en réponse à un échelon de tension")
plt.legend()
plt.show()
```



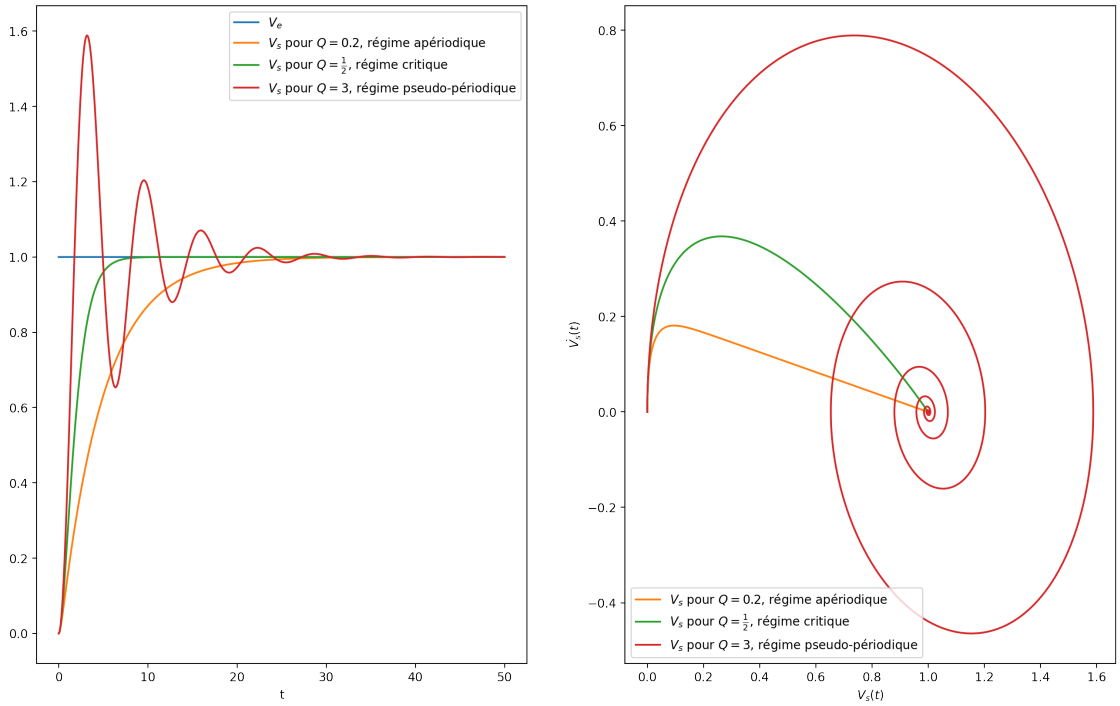

[43]:
```
ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_e} (t)$')
plt.title(r"Portraits de phase pour un circuit RLC")
plt.show()
```




Pour différents régimes :

[44]:
```
# Régime apériodique :
T,Ve,Vs1,U1 = RLC_Euler_f(0,0,0,50,1,0.2,0,100000)
# Régime critique :
T,Ve,Vs2,U2 = RLC_Euler_f(0,0,0,50,1,1/2,0,100000)
# Régime pseudo-périodique :
T,Ve,Vs3,U3 = RLC_Euler_f(0,0,0,50,1,3,0,100000)

ratio = 2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.subplot(121)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs1,label=r'$ V_s$ pour $Q=0.2$, régime apériodique')
plt.plot(T,Vs2,label=r'$ V_s$ pour $Q=\frac{1}{2}$, régime critique')
plt.plot(T,Vs3,label=r'$ V_s$ pour $Q=3$, régime pseudo-périodique')
plt.xlabel("t")
plt.legend()

plt.subplot(122)
plt.plot(Vs1,U1,label=r'$ V_s$ pour $Q=0.2$, régime apériodique', color = 'C1')
plt.plot(Vs2,U2,label=r'$ V_s$ pour $Q=\frac{1}{2}$, régime critique', color =␣
↪'C2')
plt.plot(Vs3,U3,label=r'$ V_s$ pour $Q=3$, régime pseudo-périodique', color =␣
↪'C3')
plt.xlabel(r'$V_s (t)$')
```

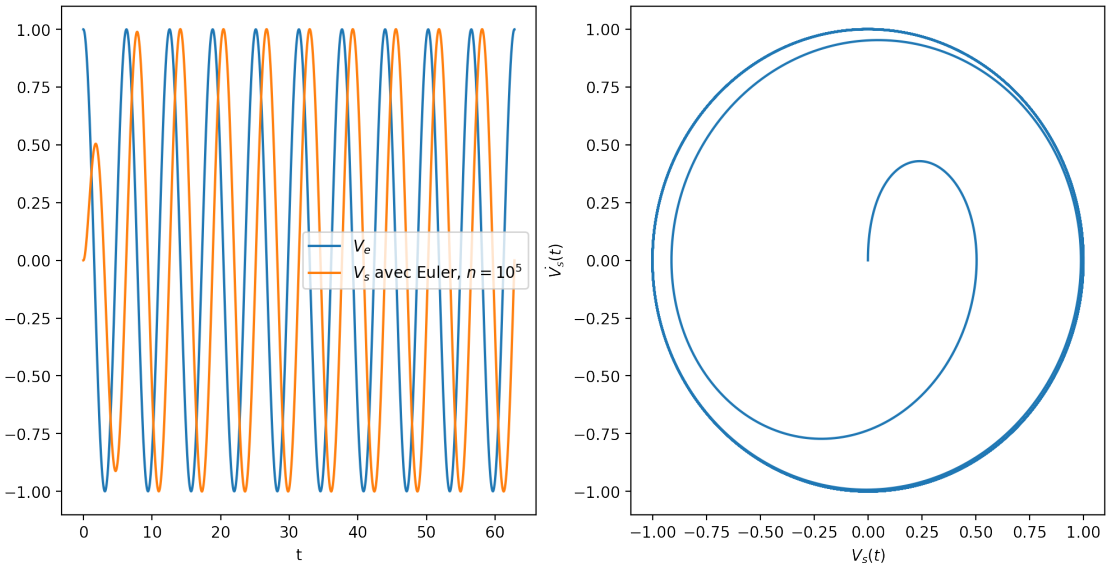
```
plt.ylabel(r'$\dot{V_s} (t)$')
plt.legend()
plt.show()
```




**Q 10 :** Régime transitoire en réponse à une excitation sinusoïdale : même question, mais pour $\omega \neq 0$ (cela représente des cas que nous ne savons pas résoudre théoriquement !).

[45]:
```
# Paramètres :
x=1
Q=1
n=int(1e5)

# On observe p périodes
p = 10
tf = p*2*np.pi/x

T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(10*ratio,5*ratio),dpi = 200)
plt.subplot(121)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$ avec Euler, $n = 10^5$')
plt.xlabel("t")
plt.legend()

plt.subplot(122)
```

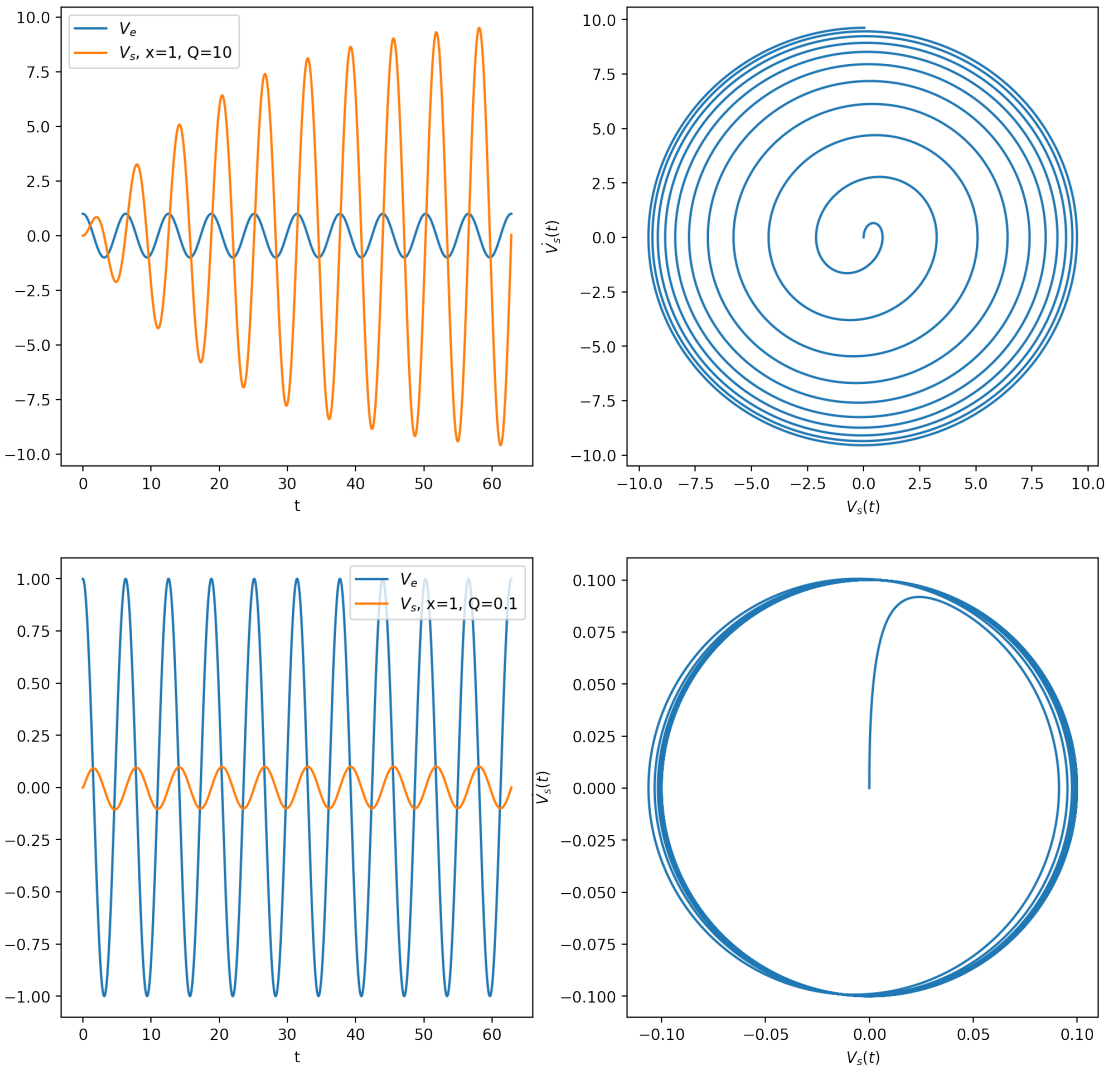
```
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

plt.show()
```




Pas très passionant ! On se trouve ici pas loin de la résonnance, proche du régime critique, régime qui permet d'atteindre le plus rapidement le régime permanent. Ici l'attracteur n'est pas un point, mais un cercle… Cherchons des cas plus amusants :

[46]:
```
ratio = 1.5 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,8*ratio),dpi = 200)

# Paramètres :
n=int(1e5)

# On observe p périodes
p = 10

# Graphe 1
x=1
tf = p*2*np.pi/x
Q=10

plt.subplot(221)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, x=1, Q=10 ')
plt.xlabel("t")
plt.legend()
```

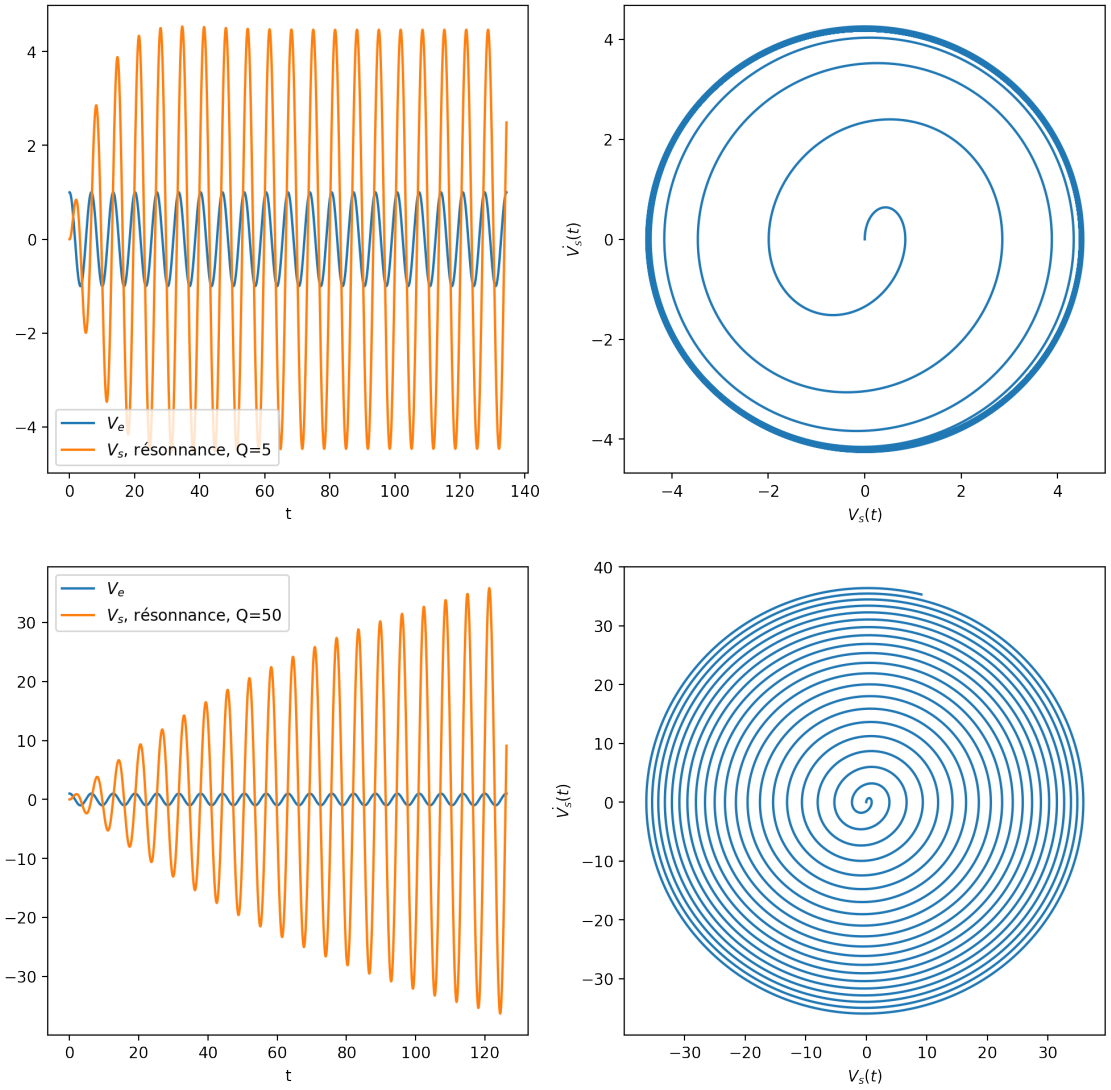
```
plt.subplot(222)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

# Graphe 2
x=1
tf = p*2*np.pi/x
Q=0.1

plt.subplot(223)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, x=1, Q=0.1 ')
plt.xlabel("t")
plt.legend()

plt.subplot(224)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

plt.show()
```




Si on change le facteur de qualité, l'amplitude finale du signal en sortie du filtre est modifiée ( elle vaut effectivement Q pour x=1 !). Bon, toujours pas très fun…écartons nous de x=1 pour se placer exactement à la pulsation de résonnance, en avec un faceur de qualité élevé :

[47]:
```
ratio = 1.5 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,8*ratio),dpi = 200)

# Paramètres :
n=int(1e5)

# On observe p périodes
p = 20

# Graphe 1
Q=5
x=sqrt(1-1/(2*Q^2))
tf = p*2*np.pi/x
```

```
plt.subplot(221)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, résonnance, Q=5 ')
plt.xlabel("t")
plt.legend()

plt.subplot(222)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

# Graphe 2
Q=50
x=sqrt(1-1/(2*Q^2))
tf = p*2*np.pi/x

plt.subplot(223)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, résonnance, Q=50 ')
plt.xlabel("t")
plt.legend()

plt.subplot(224)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

plt.show()
```

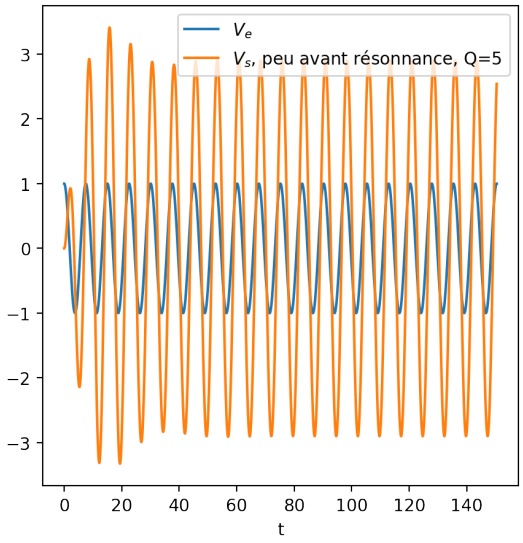


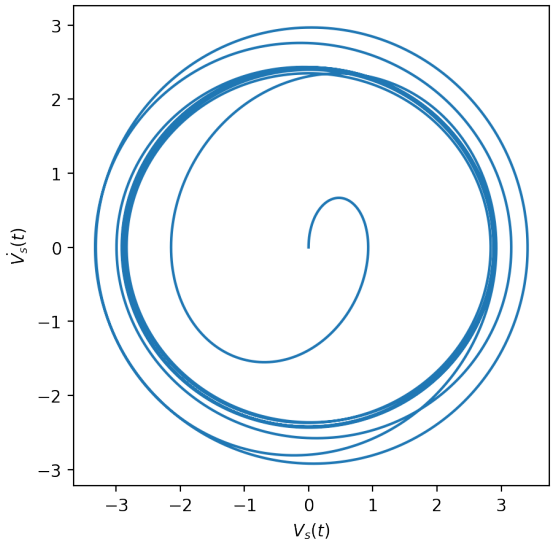

C'est le cas d'un oscillateur forçé à sa fréquence de résonnance (le moyen le pus efficace de le faire osciller…). Désaccordons légerement ce système, en abaissant légerement la fréquence d'excitation :

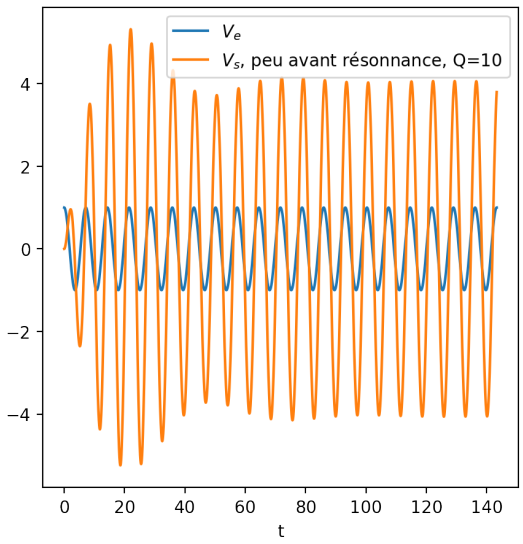
[48]:
```
ratio = 2.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(5*ratio,8*ratio),dpi = 200)

# Paramètres :
n=int(1e5)

# On observe p périodes
p = 20


# Graphe 1
Q=5
```

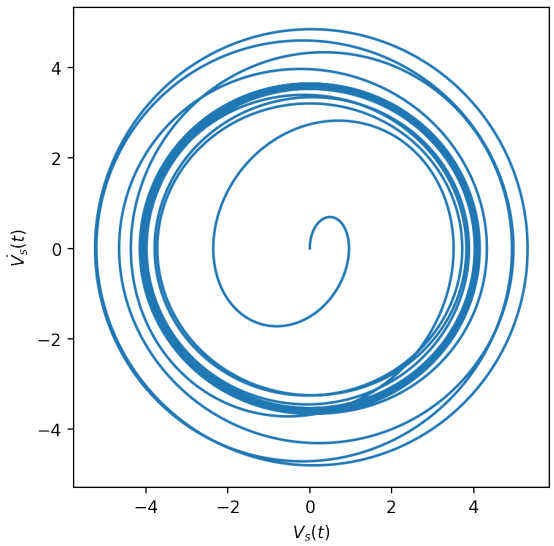
```
x=sqrt(1-1/(2*Q^2))-0.1
tf = p*2*np.pi/x
plt.subplot(321)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, peu avant résonnance, Q=5 ')
plt.xlabel("t")
plt.legend()

plt.subplot(322)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')


# Graphe 2
Q=10
x=sqrt(1-1/(2*Q^2))-0.1
tf = p*2*np.pi/x

plt.subplot(323)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, peu avant résonnance, Q=10 ')
plt.xlabel("t")
plt.legend()

plt.subplot(324)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

# Graphe 3
Q=50
x=sqrt(1-1/(2*Q^2))-0.1
tf = p*2*np.pi/x

plt.subplot(325)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, peu avant résonnance, Q=50 ')
plt.xlabel("t")
plt.legend()

plt.subplot(326)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
```

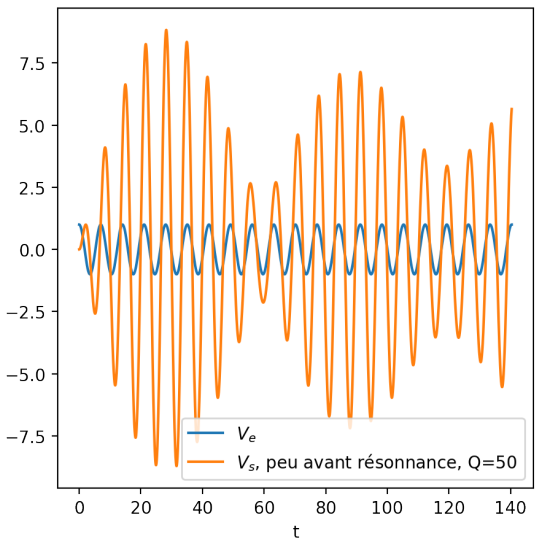
```
plt.ylabel(r'$\dot{V_s} (t)$')

plt.show()
```

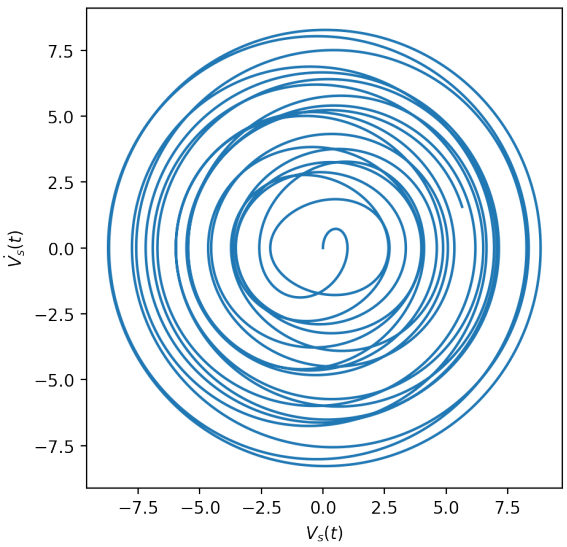

$V_e$
$V_s$, peu avant résonnance, Q=5
t


$\dot{V_s}(t)$
$V_s(t)$


$V_e$
$V_s$, peu avant résonnance, Q=10
t


$\dot{V_s}(t)$
$V_s(t)$

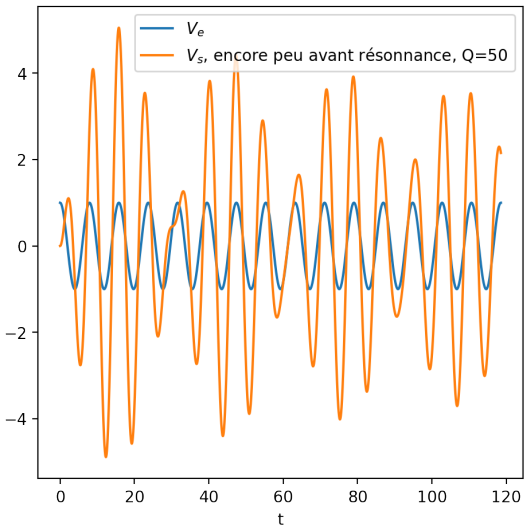

$V_e$
$V_s$, peu avant résonnance, Q=50
t

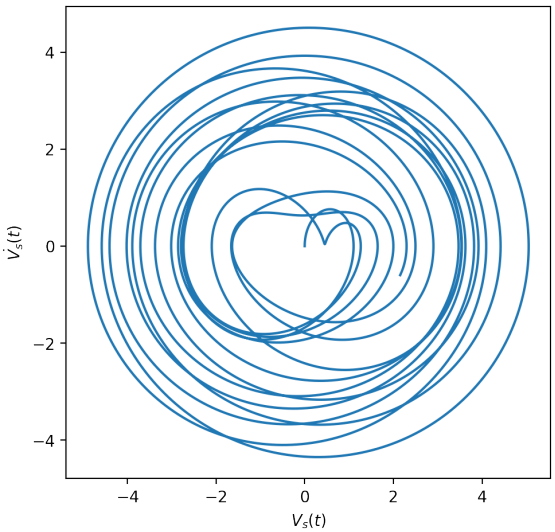

$\dot{V_s}(t)$
$V_s(t)$

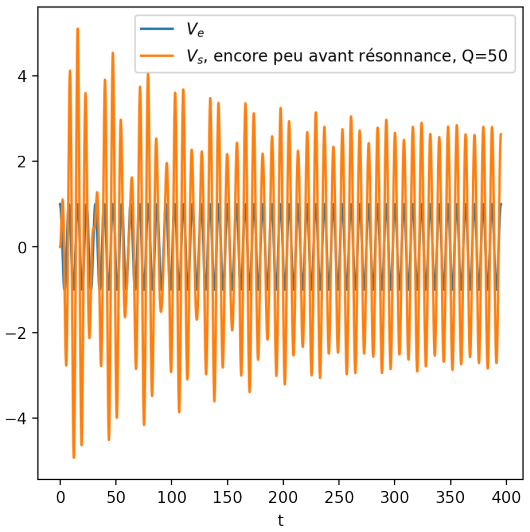
Hooo les jolis battements !!!! On reconnait bien evidemment un système avec deux fréquences (celle propre de l'oscillateur, et celle de forçage), proches l'une de l'autre. On peut aussi faire le lien avec le TP de TPC 1 sur l'oscillateur mécanique forçé, où on a pu observer de tels comportements. Et quelle est la fréquence de ces battements ? Ca doit surement avoir à faire avec la différence de ces deux fréquences… Vérifions :

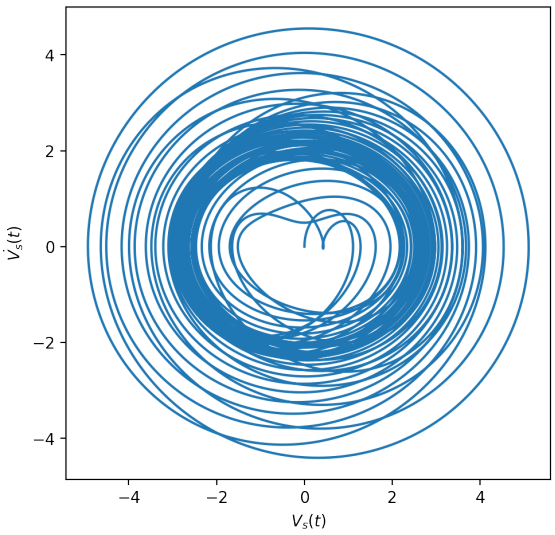
[49]:
```
ratio = 1.5 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,8*ratio),dpi = 200)

# Paramètres :
n=int(1e5)

# On observe p périodes
p = 15

Q=50
x=sqrt(1-1/(2*Q^2))-0.2
tf = p*2*np.pi/x

plt.subplot(221)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, encore peu avant résonnance, Q=50 ')
plt.xlabel("t")
plt.legend()

plt.subplot(222)
plt.plot(Vs,U)
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

# On observe p périodes
p = 50

Q=50
x=sqrt(1-1/(2*Q^2))-0.2
tf = p*2*np.pi/x

plt.subplot(223)
T,Ve,Vs,U = RLC_Euler_f(0,0,0,tf,1,Q,x,n)
plt.plot(T,Ve,label=r'$ V_e$')
plt.plot(T,Vs,label=r'$ V_s$, encore peu avant résonnance, Q=50 ')
plt.xlabel("t")
plt.legend()

plt.subplot(224)
plt.plot(Vs,U)
```

```
plt.xlabel(r'$V_s (t)$')
plt.ylabel(r'$\dot{V_s} (t)$')

plt.show()
```




[ ]: