```python
# -*- coding: utf-8 -*-
"""
Created on Fri Sep  2 14:36:09 2022

@author: fabrice
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from matplotlib import rc

fig , axes = plt.subplots(nrows = 2 , ncols = 2 , figsize = (9,9))

font_properties = {'size' : 14}
rc('font' , **font_properties)

#équation: x''(t)-ε(1-x**2)x'(t) +x = Asin(wt)
#paramètres de l'équation A=amplitude du forçage, w=fréquence du forçage
A , w = 1 , 0.5

#paramètre d'amortissement ε = e plusieurs valeurs
e1 , e2  = 0.5  , 5
vect_e = e1 , e2

#conditions initiales
x0 , dx0 = 0.1 , 0
vect_init = x0 , dx0

#fichier base de temps
tmin , tmax  = 0 , 100
deltat = 1000
t = np.linspace(tmin , tmax, deltat)


j = 0 #compteur pour placer les graphiques

#boucle qui fait faire le calcul de x pour les deux valeurs de e
for e in vect_e:

    #fonction qui renvoie dx/dt et d2x/dt2 à l'instant t
    #on pose x1=x et x2=dx/dt
    def derive(t , y):
        x1 , x2 = y
        dx1dt = x2
        dx2dt = e*(1-x1**2)*x2  - x1 + A*np.sin(w*t)

        return dx1dt , dx2dt


    #intégration numérique de l'edo
    solution = solve_ivp(derive , (tmin, tmax) , vect_init , dense_output = True)

    #extraction des solutions x et sa dérivée sur la base des t
```

```
    x , dx = solution.sol(t)

    #dessine les graphiques pour chaque e: x(t) et dx/dt versus x
    ax = axes[j, 0]
    ax.plot(t , x , c = 'k' )
    ax.set_title('x(t)  ε='+str(e))
    ax.set_xlabel('temps (s)')
    ax.set_ylabel('x (a.u)')
    ax.grid()

    ax = axes[j , 1]
    ax.plot(x , dx , c = 'k' )
    ax.set_title('Espace des phases  ε='+str(e))
    ax.set_xlabel('x(t) (a.u)')
    ax.set_ylabel('d(x)/dt (a.u,)')
    ax.grid()

    j+=1

#affiche les graphiques
plt.suptitle('Oscillateur de Van der Pol')
fig.tight_layout()
plt.show()
```