Feuille_info_14 : Fonction `odeint`

On considère que les importations suivantes ont été faites :

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as si
```

Le module `scipy.integrate` possède une fonction `odeint` qui permet comme la méthode d'Euler de résoudre des problèmes de Cauchy :

$$Y'(t) = F(Y(t), t) \qquad \text{et} \qquad Y(t_0) = Y_0$$

Aucune question sur la méthode utilisée par `odeint` *vous devez juste montrer que vous savez l'utiliser.*

Un premier exemple avec `odeint` et la solution sur $[0, 2]$ du problème : $y' = 2y + t$ avec $y(0) = 1$

```
def F(y, t):
    return 2*y + t

t = np.linspace(0, 2, 100)        # contient [t_0, t_1, ... , t_n]
y = si.odeint(F ,[1], t)          # contient [y_0, y_1, ... , y_n]

plt.plot(t,y)
plt.show()
```

Ex 1 : On considère les problèmes différentiels suivants :

1) sur $[1, 4]$, $y'(t) = 2y(t) + 4t$ et $y(1) = 3$
2) sur $[0, 3]$, $2y'(t) + 2y^2(t) = 5$ et $y(0) = 1$
3) sur $[0, 2]$, $y'(t) = \sin(y(t))$ et $y(0) = 4$
4) sur $[1, 4]$, $y'(t) + 2ty(t) = 4t$ et $y(1) = 3$
5) sur $[0, 10]$, $y'(t) = \exp(-y(t)) + 4t$ et $y(0) = 1$
6) sur $[1, 3]$, $y'(t) = 2y(t) + 4t$ et $y(2) = -2$

Compléter le programmes suivants permettant d'avoir une solution approchée :

1)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[    ], t)
```

2)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[   ], t)
```

3)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[   ], t)
```

4)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[   ], t)
```

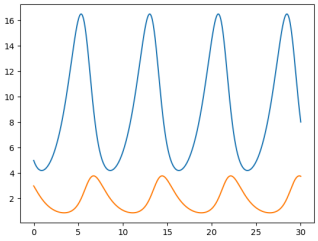
5)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[   ], t)
```

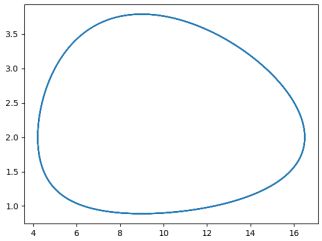
6)
```
def F(y, t):
  return

t = np.linspace(   ,   , 1000)
y = si.odeint(F ,[   ], t)
```

Un deuxième exemple avec `odeint` et la solution sur $[0,10]$ du problème :

$$x(0)=5\ ,\quad y(0)=3\quad \text{et}\quad \begin{cases} \dfrac{dx}{dt}(t) &= x(t)\,(a-b\,y(t)) \\ \dfrac{dy}{dt}(t) &= -y(t)\,(c-d\,x(t)) \end{cases}$$

```
def F(Y,t):
    a, b, c, d = 0.8, 0.4, 0.9 , 0.1
    x = Y[0]
    y = Y[1]
    return [(a-b*y*x, -(c-d*x)*y]

t = np.linspace(t0, t1, 200)
Y = si.odeint(F ,[5, 3], t)        # contient [[x_0, y_0], [x_1, y_1], ...., [x_n, y_n]]
x = Y[:, 0]
y = Y[:, 1]

plt.plot(t, x)   #  x en fonction de t
plt.plot(t, y)   #  y en fonction de t



plt.plot(x, y)   #  y en fonction de x
```





Ex 2 :

1) pour $t\in[0,3]$, $\begin{cases} x'(t)=2x(t)+y(t)+t \\ y'(t)=3x(t)-y(t)-2t \end{cases}$ et $\begin{cases} x(0)=5 \\ y(0)=6 \end{cases}$

2) pour $t\in[0,10]$, $\begin{cases} x'(t)=2x(t)y(t)+y(t) \\ y'(t)=3x(t)-x(t)y(t) \end{cases}$ et $\begin{cases} x(0)=1 \\ y(0)=2 \end{cases}$

3) sur $[0,1]$, $y''(t)+(y'(t))^2+2y(t)=4$ et $y(0)=3$ et $y'(0)=0$

4) sur $[1,4]$, $y''(t)+ty'(t)+y(t)=4t$ et $y(1)=3$ et $y'(1)=0$

5) sur $[0,4]$, $y''(t)=2y(t)y'(t)$ et $y(0)=1$ et $y'(0)=0$

6) sur $[0,4]$, $y''(t)-2y'(t)=-te^t$ et $y(0)=1$ et $y'(0)=0$

Compléter le programmes suivants permettant d'avoir une solution approchée des problèmes ci-dessus :

1)
```
def F(Y, t):
    x = Y[0]
    y = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

2)
```
def F(Y, t):
    x = Y[0]
    y = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

3)
```
def F(Y, t):
    y = Y[0]
    v = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

4)
```
def F(Y, t):
    y = Y[0]
    v = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

5)
```
def F(Y, t):
    y = Y[0]
    v = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

6)
```
def F(Y, t):
    y = Y[0]
    v = Y[1]
    return

t = np.linspace(   ,    , 1000)
Y = si.odeint(F ,[     ], t)
```

Ex 3 : Reprendre **Ex 1** et **Ex 2** et comparer le résultat obtenu avec la méthode d'Euler et celui obtenu avec `odeint`.

Annexe

L'équation du mouvement du pendule simple est :

$$\theta''(t) + \omega_0^2 \sin\big(\theta(t)\big) = 0 \qquad \text{en posant : } \omega_0^2 = \frac{g}{\ell}$$

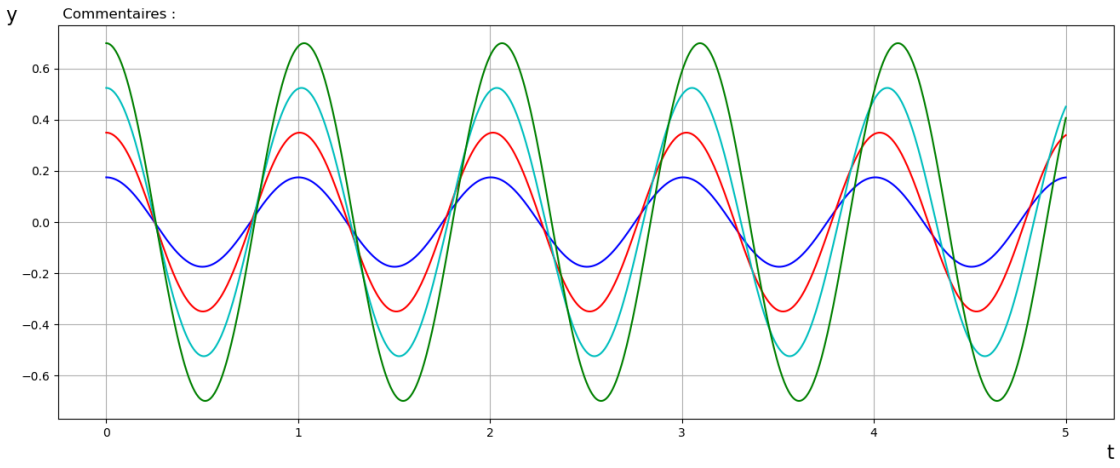
La résolution exacte de cette équation est hors de notre portée. On se contente d'une résolution approchée pour des petites oscillations, en faisant l'approximation : $\sin(\theta) \approx \theta$ on obtient :

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

Le but de cette annexe est d'illustrer cette approximation.

La méthode numérique de `odeint` permet d'étudier l'équation $\theta''(t) + \omega_0^2 \sin(\theta(t)) = 0$.

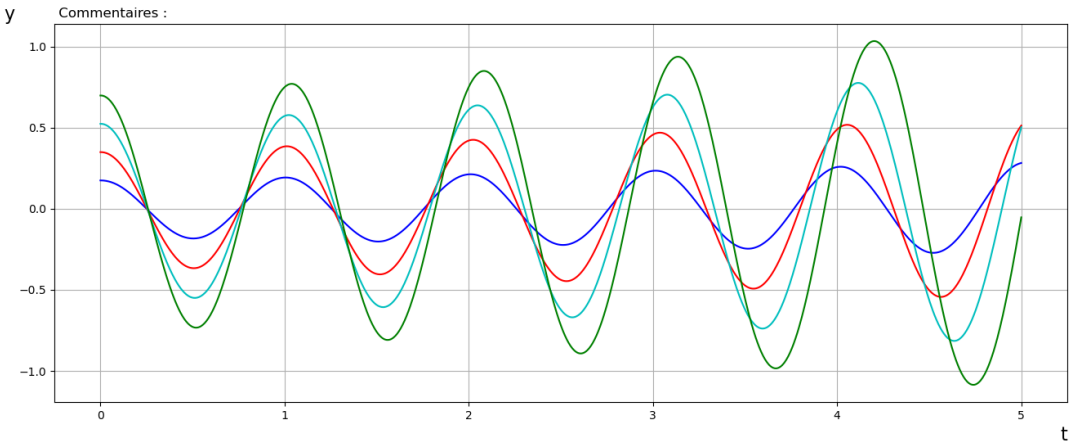
```
def derive(Y, t):
    y, v = Y[0], Y[1]
    return [v,  - w0**2 * np.sin(y)]

T = 5
t = np.linspace(0, T, 1000)
w0 = 2*np.pi

A1 = 10/360*2*np.pi              # Ampitude de 10 deg
Y1i = [A1, 0]                    # Condition initiale
Y1 = odeint(derive, Y1i, t)      # Méthode de la fonction odeint
A2 = 20/360*2*np.pi              # Même chose avec une amplitude de 20 deg
Y2i = [A2, 0]
Y2 = odeint(derive, Y2i, t)
A3 = 30/360*2*np.pi              # Même chose avec une amplitude de 30 deg
Y3i = [A3, 0]
Y3 = odeint(derive, Y3i, t)
A4 = 40/360*2*np.pi              # Même chose avec une amplitude de 40 deg
Y4i = [A4, 0]
Y4 = odeint(derive, Y4i, t)

plt.figure('Avec odeint', figsize = (16, 6))
plt.plot(t, Y1[:,0], "b-")
plt.plot(t, Y2[:,0], "r-")
plt.plot(t, Y3[:,0], "c-")
plt.plot(t, Y4[:,0], "g-")
plt.xlabel('t', fontsize=16, loc = 'right')
plt.ylabel('y', fontsize=16, loc = 'top', rotation = 0)
plt.title(' Commentaires :  ')
plt.grid()
plt.show()
```



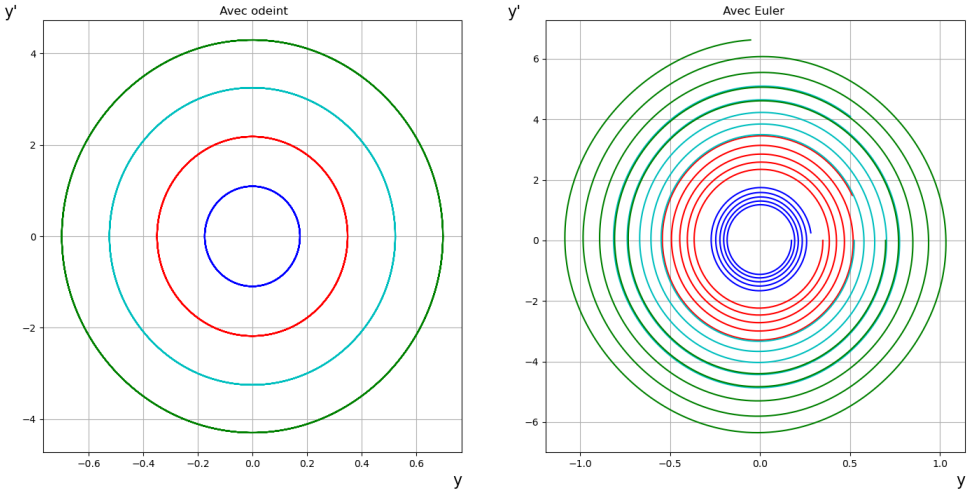
Commenter ces courbes :

On a utilisé la méthode de `odeint` qui est bien plus précise que la méthode d'Euler.
Voyons ci-dessous ce que donne la méthode d'Euler.

```
def methode_d_Euler(F, y0, y1, t0, tf, N):
    dt = (tf-t0)/N
    t, y, z = [ t0 ], [ y0 ], [ y1 ]
    for k in range(N):
        t.append(t[k] + dt)
        y.append(y[k] + dt*F([y[k],z[k]], t[k])[0])
        z.append(z[k] + dt*F([y[k],z[k]], t[k])[1])
    return np.array([[y[k],z[k]] for k in range(N)])

F = derive
N = 1000
A1 = 10/360*2*np.pi              # Ampitude de 10 deg
Y1 = methode_d_Euler(F, A1, 0, 0, 5, N)
A2 = 20/360*2*np.pi              # Même chose avec une amplitude de 20 deg
Y2 = methode_d_Euler(F, A2, 0, 0, 5, N)
A3 = 30/360*2*np.pi              # Même chose avec une amplitude de 30 deg
Y3 = methode_d_Euler(F, A3, 0, 0, 5, N)
A4 = 40/360*2*np.pi              # Même chose avec une amplitude de 40 deg
Y4 = methode_d_Euler(F, A4, 0, 0, 5, N)

plt.plot(t, Y1[:,0], "b-")
plt.plot(t, Y2[:,0], "r-")
plt.plot(t, Y3[:,0], "c-")
plt.plot(t, Y4[:,0], "g-")
plt.xlabel('t', fontsize=16, loc = 'right')
plt.ylabel('y', fontsize=16, loc = 'top', rotation = 0)
plt.grid()
plt.show()
```



Les portraits de phase permet de bien visualiser la différence des deux méthodes numériques :



Si vous avez le temps vous pouvez ajouter un terme d'amortissement : $\theta''(t) + 2\lambda\theta'(t) + \omega_0^2 \sin\big(\theta(t)\big) = 0$