TP 11

Equations différentielles
Sources :

Proposition de corrigé

Activité 1 : Equations différentielles et pendule simple

En préambule :

```
from math import sqrt, pi, sin, cos
from numpy import array, linspace
from matplotlib import pyplot as pl
from scipy.integrate import odeint
```

Il est utile de fixer en préambule les différentes constantes utilisées dans le TP.

```
alpha = 0.5
h = 0
omega0 = sqrt(2*pi)
omega = 1

a = 0
b = 10
```

On copie du cours :

```
def euler(F, a, b, y0, h):
    """Solution de y'=F(y,t) sur [a,b], y(a) = y0, pas h"""
    y = y0
    t = a
    y_list = [y0] # la liste des valeurs renvoyées
    t_list = [a] # la liste des temps
    while t+h <= b:
        # Variant : floor((b-t)/h)
        # Invariant : au tour k, y_list = [y_0,...,y_k], t_list = [t_0,...,t_k]
        y = y + h * F(y, t)
        y_list.append(y)
        t = t + h
        t_list.append(t)
    return t_list, y_list
```

Q 1 : La variable est

$$\Theta(t)=\begin{pmatrix}\theta(t)\\ \theta'(t)\end{pmatrix}$$

et les fonctions demandées aux questions 1 et 12 sont

$$F:\begin{array}{ccc}\mathbb{R}^2\times\mathbb{R} & \to & \mathbb{R}^2\\ \begin{pmatrix}x\\y\end{pmatrix},t & \mapsto & \begin{pmatrix}y\\ -\alpha y-\omega_0^2\sin(x)\end{pmatrix}\end{array},$$

$$F_{sf}:\begin{array}{ccc}\mathbb{R}^2\times\mathbb{R} & \to & \mathbb{R}^2\\ \begin{pmatrix}x\\y\end{pmatrix},t & \mapsto & \begin{pmatrix}y\\ -\omega_0^2\sin(x)\end{pmatrix}\end{array},$$

$$F_{po}:\begin{array}{ccc}\mathbb{R}^2\times\mathbb{R} & \to & \mathbb{R}^2\\ \begin{pmatrix}x\\y\end{pmatrix},t & \mapsto & \begin{pmatrix}y\\ -\omega_0^2 x\end{pmatrix}\end{array},$$

$$F_f:\begin{array}{ccc}\mathbb{R}^2\times\mathbb{R} & \to & \mathbb{R}^2\\ \begin{pmatrix}x\\y\end{pmatrix},t & \mapsto & \begin{pmatrix}y\\ \cos(\omega t)-\alpha y-\omega_0^2\sin(x)\end{pmatrix}\end{array}.$$

Q 2 : L'ensemble des solutions est

$$\left\{\begin{array}{ccc}\mathbb{R} & \to & \mathbb{R}\\ t & \mapsto & \lambda\cos(\omega_0 t)+\mu\sin(\omega_0 t)\end{array}\;\middle|\;\lambda,\mu\in\mathbb{R}\right\}.$$

Avec la condition initiale $\theta'(0)=0s^{-1}$, on obtient la solution

$$\begin{array}{ccc}\mathbb{R} & \to & \mathbb{R}\\ t & \mapsto & \theta_0\cos(\omega_0 t)\end{array}$$

Q 3 :

```
def Fpo (Theta, t) :
    """(x,y)-> (y,-omega0**2*x)"""
    x = Theta[0]
    y = Theta[1]
    return array([y,-omega0**2*x])
```

Q 4 :

```
def trace_po (th0, n, nom_de_fichier) :
    """Tracé des oscillations, approximation des petites oscillations
    CI : theta(0) = th0, n segments"""
    pl.clf()
    pl.grid()
    pl.title('Oscillations au cours du temps,
        $\\theta(0)='+str(th0)+"$, $n="+str(n)+"$")
    pl.xlabel('$t$')
    pl.ylabel('$\\theta(t)$')
    y0 = array([th0,0])
    t_list, Y = euler(Fpo,a,b,y0,10/n)
    y_list = [theta(x,th0) for x in t_list]
    pl.plot(t_list,y_list,'r',label="Solution exacte")
    y_list = [y[0] for y in Y]
    pl.plot(t_list,y_list,'b',label="Méthode d'Euler")
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Q 5 :

```
def rk4(F, a, b, y0, h):
    """Solution de y'=F(y,t) sur [a,b], y(a) = y0, pas h"""
    y = y0
    t = a
    y_list = [y0] # la liste des valeurs renvoyées
    t_list = [a]
```

```
    while t+h <= b:
        # Variant : floor((b-t)/h)
        # Invariant : au tour k, y_list = [y_0,...,y_k], t_list = [t_0,...,t_k]
        k1 = F(y, t)
        k2 = F(y + (h/2)*k1, t + h/2)
        k3 = F(y + (h/2)*k2, t + h/2)
        k4 = F(y + h*k3, t + h)
        y = y + h * (k1+2*k2+2*k3+k4)/6 # surtout pas += !
        y_list.append(y)
        t += h
        t_list.append(t)
    return t_list, y_list
```

Q 6 :

```
def trace_po_rk4 (th0, n, nom_de_fichier) :
    """Tracé des oscillations, approximation des petites oscillations
    CI : theta(0) = th0, n segments"""
    pl.clf()
    pl.grid()
    pl.title('Oscillations au cours du temps,
        $\\theta(0)='+str(th0)+"$, $n="+str(n)+"$")
    pl.xlabel('$t$')
    pl.ylabel('$\\vartheta(t)$')
    y0 = array([th0,0])
    t_list, Y = euler(Fpo,a,b,y0,10/n)
    y_list = [theta(x,th0) for x in t_list]
    pl.plot(t_list,y_list,'r',label="Solution exacte")
    y_list = [y[0] for y in Y]
    pl.plot(t_list,y_list,'b',label="Euler")
    _, Y = rk4(Fpo,a,b,y0,10/n)
    y_list = [y[0] for y in Y]
    pl.plot(t_list,y_list,'g',label="rk4")
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Même avec $n = 50$, la solution approchée donnée par la méthode de Runge-Kutta est indiscernable de la soultion exacte, alors que l'approximation donnée par la méthode d'Euler s'en éloigne très rapidement.

Q 7 :

```
def Fsf (Theta, t) :
    """(x,y)-> (y,-omega0**2*sin(x))"""
    x = Theta[0]
    y = Theta[1]
    return array([y,-omega0**2*sin(x)])
```

Q 8 :

```
def approx_po (th0, n, nom_de_fichier) :
    """Approximation des petites oscillations vs simulation par la méthode d'Euler
    CI : theta(0) = th0, n segments"""
    pl.clf()
    pl.grid()
    pl.title('Oscillations au cours du temps,
        $\\theta(0)='+str(th0)+"$, $n="+str(n)+"$")
    pl.xlabel('$t$')
    pl.ylabel('$\\theta(t)$')
    y0 = array([th0,0])
    t_list, Y = euler(Fsf,a,b,y0,10/n)
    y_list = [theta(x,th0) for x in t_list]
    pl.plot(t_list,y_list,'r',label="Solution exacte des petites oscillations")
    y_list = [y[0] for y in Y]
    pl.plot(t_list,y_list,'g',label="Méthode d'Euler")
```



```
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Pour $\theta_0 = 2$, les deux solutions ont l'air périodiques mais n'ont pas la même période, celle donnée par la méthode d'Euler a une période plus longue.

Q 9 : La somme des distances entre deux pics consécutifs est la distance entre le premier et le dernier pic (sommation télescopique).

```
def periode(L):
    """Période du tableau L"""
    n = len(L)
    c = 0 # nombre de pics trouvés
    for i in range(1,n-1):
        if L[i]>L[i-1] and L[i]>L[i+1] :
            c = c+1
            #L[i] est un pic
            if c == 1 :
                premier = i
                # indice du premier pic
            dernier = i
            # indice du dernier pic
    if c >= 2 :
        # Au moins deux pics
        return (dernier - premier) / (c-1)
```

Q 10 :

```
def periode_pendule(n) :
    """Tableau des estimations des périodes du pendule, n points"""
    a = 0
    b = 10*2*pi/omega0 # Dix périodes dans les petites oscillations.
    h = b / 1000
    T_list = []
    for k in range(1,n+1) :
        y0 = array([(k*pi)/(2*n),0])
        _,Y = rk4(Fsf,a,b,y0,h)
        L = [y[0] for y in Y]
        T_list.append(periode(L)*h)
    return T_list
```

Q 11 :

```
def trace_periode (n, nom_de_fichier) :
    x = [(k*pi)/(2*n) for k in range(1,n+1)]
    y = periode_pendule(n)
    pl.clf()
    pl.grid()
    pl.title("Période du pendule simple en fonction de l'angle initial")
    pl.xlabel('$\\theta(0)$')
    pl.ylabel('Période en $s$')
    pl.plot(x,y)
    pl.savefig(nom_de_fichier)
```

Q 12 : Cf. Q1.

Q 13 :

```
def Ff (Theta, t) :
    """(x,y)-> (y, -alpha*y-omega0**2*sin(x)+cos(omega*t))"""
    x = Theta[0]
    y = Theta[1]
    return array([y, -alpha*y-omega0**2*sin(x)+cos(omega*t)])
```

Q 14 :

```
def trace_trajectoire_f(th0,thp0,n,nom_de_fichier):
    """Trajectoire du pendule forcé, avec frottements, par la méthode d'Euler
    CI : theta(0)=th0, theta'(0)=thp0, n segments"""
    pl.clf()
    pl.grid()
    pl.title('Oscillations forcées avec frottements,
        $\\theta(0)='+str(th0)+"$,$\\theta'(0)="+str(thp0)+"$, $n="+str(n)+"$")
    pl.xlabel('$t$')
    pl.ylabel('$\\theta(t)$')
    y0 = array([th0,thp0])
    t_list, Y = euler(Ff,a,b,y0,(b-a)/n)
    y_list = [y[0] for y in Y]
    pl.plot(t_list,y_list,'b',label="Méthode d'Euler")
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Q 15 :

```
def trace_phase_f(th0,thp0,n,nom_de_fichier):
    """Portrait de phase du pendule forcé, avec frottements, par la méthode d'Euler
    CI : theta(0)=th0, theta'(0)=thp0, n segments"""
    pl.clf()
    pl.grid()
    pl.title('Oscillations forcées avec frottements,
        $\\theta(0)='+str(th0)+"$,$\\theta'(0)="+str(thp0)+"$, $n="+str(n)+"$")
    pl.xlabel("$\\theta(t)$")
    pl.ylabel("$\\theta'(t)$")
    y0 = array([th0,thp0])
    _, Y = euler(Ff,a,b,y0,(b-a)/n)
    t_list = [y[0] for y in Y]
    y_list = [y[1] for y in Y]
    pl.plot(t_list,y_list,'b',label="Méthode d'Euler")
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Q 16 :

```
def odeint_f(th0, thp0, n):
    """Solution de l'équadiff Pf par odeint"""
    t_list = linspace(a, b, n+1)
    y0 = array([th0, thp0])
    y_list = odeint(Ff, y0, t_list)
    return t_list, y_list
```

Q 17 :

```
def trace_trajectoire_odeint(t_list,y_list,nom_de_fichier):
    """Trajectoire du pendule forcé, avec frottements, par odeint
    CI : theta(0)=th0, theta'(0)=thp0, n segments"""
    pl.clf()
    pl.grid()
    th0 = y_list[0][0]
    thp0 = y_list[0][1]
    pl.title('Oscillations forcées avec frottements,
        $\\theta(0)='+str(th0)+"$,$\\theta'(0)="+str(thp0)+"$")
    pl.xlabel('$t$')
    pl.ylabel('$\\theta(t)$')
    theta_list = [y[0] for y in y_list]
    pl.plot(t_list,theta_list,'b',label="Solution d'odeint")
    pl.legend()
    pl.savefig(nom_de_fichier)
```

Q 18 :

Cycle 01


```
def trace_phase_odeint(t_list,y_list,nom_de_fichier):
    """Portrait de phase du pendule forcé, avec frottements, par odeint
    CI : theta(0)=th0, theta'(0)=thp0, n segments"""
    pl.clf()
    pl.grid()
    th0 = y_list[0][0]
    thp0 = y_list[0][1]
    pl.title('Oscillations forcées avec frottements,
        $\\theta(0)='+str(th0)+"$,$\\theta'(0)="+str(thp0)+"$")
    pl.xlabel("$\\theta(t)$")
    pl.ylabel("$\\theta'(t)$")
    theta_list = [y[0] for y in y_list]
    thetap_list = [y[1] for y in y_list]
    pl.plot(theta_list,thetap_list,'b',label="Solution d'odeint")
    pl.legend()
    pl.savefig(nom_de_fichier)
```