```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar 10 06:07:45 2014

@author: stephane
"""

import numpy as np

import matplotlib.pyplot as pypl

from scipy.integrate import odeint

# Exo 1 : fait


def euler(F, a, y0, b, n):
    les_yk = [y0]         # la liste des valeurs calculées
    t = a                 # le temps du dernier calcul
    h = float(b-a) / n  # le pas
    dernier = y0          # la dernière valeur calculée
    for i in range(n):
        suivant = dernier + h*F(t, dernier) # le nouveau terme
        les_yk.append(suivant)  # on le place à la fin des valeurs calculées
        t = t+h                 # le nouveau temps
        dernier = suivant       # et on met à jour le dernier terme calculé
    return les_yk  # c'est fini


# Exo 2 :

def heun(F, a, y0, b, n):
    les_yk = [y0]         # la liste des valeurs calculées
    t = a                 # le temps du dernier calcul
    h = float(b-a) / n  # le pas
    dernier = y0          # la dernière valeur calculée
    for i in range(n):
        eul = dernier + h*F(t, dernier) # ce que propose Euler
        suivant = dernier + h/2*(F(t, dernier) + F(t+h, eul)) # le nouveau terme
        les_yk.append(suivant)  # on le place à la fin des valeurs calculées
        t = t+h                 # le nouveau temps
        dernier = suivant       # et on met à jour le dernier terme calculé
    return les_yk

"""
>>> def f1(t, z): # attention à l'ordre des arguments !
...     return z
...
>>> heun(f1, 0, 1, 1, 2)
[1, 1.625, 2.640625]
"""
def f0(z, t): # attention à l'ordre des arguments !
    return z
def f1(t, z): # attention à l'ordre des arguments !
    return z

# Exo 3 :

def RK4(F, a, y0, b, n):
    les_yk = [y0]         # la liste des valeurs calculées
    t = a                 # le temps du dernier calcul
    h = float(b-a) / n  # le pas
    dernier = y0          # la dernière valeur calculée
    for i in range(n): # NE PAS S'AMUSER À PRENDRE D'AUTRES NOTATIONS !
        alpha = dernier + h/2*F(t, dernier)
        beta = dernier + h/2*F(t+h/2, alpha)
        gamma = dernier + h*F(t+h/2, beta)
        suivant = dernier + h/6*(F(t, dernier) + 2*F(t+h/2, alpha) +
                          2*F(t+h/2, beta) + F(t+h, gamma)) # le nouveau terme
        les_yk.append(suivant)  # on le place à la fin des valeurs calculées
        t = t+h                 # le nouveau temps
        dernier = suivant       # et on met à jour le dernier terme calculé
    return les_yk

"""
>>> RK4(f1, 0, 1, 1, 2)
[1, 1.6484375, 2.71734619140625]
"""

# Exo 4 :

from math import exp
e = exp(1)

print("exp(1)=%.15f"%e)
for n in [10, 100, 1000]:
    eul = euler(f1, 0, 1, 1, n)[-1]
    heu = heun(f1, 0, 1, 1, n)[-1]
    rk4 = RK4(f1, 0, 1, 1, n)[-1]
    print("Pour n=%i :\n\t\t%.15f\t%.15f\t%.15f"%(n,eul,heu,rk4))
    print("Erreurs absolues : \n\t\t%.15f\t%.15f\t%.15f\n"\
          %(abs(eul-e), abs(heu-e), abs(rk4-e)))

"""
exp(1)=2.718281828459045
Pour n=10 :
                2.593742460100000       2.714080846608224       2.718279744135166
Erreurs absolues :
                0.124539368359045       0.004200981850821       0.000002084323879

Pour n=100 :
                2.704813829421526       2.718236862559957       2.718281828234404
Erreurs absolues :
                0.013467999037519       0.000044965899088       0.000000000224641

Pour n=1000 :
                2.716923932235896       2.718281375751763       2.718281828459025
Erreurs absolues :
                0.001357896223149       0.000000452707282       0.000000000000020
"""

t = np.linspace(0, 1, 5)
# Les temps t_k. Il en faut le même nombre que les yk, attention...

yeuler = euler(f1, 0, 1, 1, 4)
yheun = heun(f1, 0, 1, 1, 4)
yrk4 = RK4(f1, 0, 1, 1, 4)

yodeint = odeint(f0, 1, t)

pypl.plot(t, yeuler)
pypl.plot(t, yheun)
pypl.plot(t, yrk4)
pypl.plot(t, yodeint)

pypl.legend(['Euler', 'Heun', 'RK4', 'odeint'], loc = 'upper left')
pypl.title(r"Comparaison (avec un pas de $1/4$) des methodes pour $y'=y$")

pypl.savefig('comparaison-methodes.pdf')
pypl.show()
pypl.clf() # Pour effacer : penser à la suite du tp !

# Exos 5 et 6

def f_pendule(t, z):
    y, yp = z # Ack TERRIBLE pour éviter (y, yp) = (z[0], z[1]). C'est MAL...
    return np.array([yp, -np.sin(y)])
```

```
for n in [20, 100, 1000]:
    Y = euler(f_pendule, 0, np.array([0, 1.5]), 10, n)
    Yheun = heun(f_pendule, 0, np.array([0, 1.5]), 10, n)
    Yrk4 = RK4(f_pendule, 0, np.array([0, 1.5]), 10, n)

    Yarray = np.array(Y)
    Yharray = np.array(Yheun)
    Y4array = np.array(Yrk4)

    t = np.linspace(0, 10, 1+n)
    pypl.plot(t, Yarray[:, 0], linewidth=3)
    # sur chaque ligne de Y, on prend la première composante
    pypl.plot(t, Yharray[:, 0], linewidth=1)
    pypl.plot(t, Y4array[:, 0], linewidth=1)

    pypl.axhline(color='black')
    pypl.grid()
    pypl.legend(['Euler', 'Heun', 'RK4'], loc = 'lower left')
    pypl.title(r'Pendule non amorti : $n=$'+str(n))
    pypl.savefig('pendule'+str(n)+'.pdf')
    pypl.show()
    pypl.clf()


# Exo 7

n = 10000
Y = euler(f_pendule, 0, np.array([0, 2]), 30, n)
Yheun = heun(f_pendule, 0, np.array([0, 2]), 30, n)
Yrk4 = RK4(f_pendule, 0, np.array([0, 2]), 30, n)

Yarray = np.array(Y)
Yharray = np.array(Yheun)
Y4array = np.array(Yrk4)

t = np.linspace(0, 30, 1+n)
pypl.plot(t, Yarray[:, 0], linewidth=3)
pypl.plot(t, Yharray[:, 0], '--', linewidth=2)
pypl.plot(t, Y4array[:, 0], linewidth=1)

def dulepen(z, t):
    y, yp = z
    return np.array([yp, -np.sin(y)])
Yodeint = odeint(dulepen, [0, 2], t)
pypl.plot(t,Yodeint[:, 0], linewidth=4)

pypl.grid()
pypl.legend(['Euler', 'Heun', 'RK4', 'odeint'], loc = 'upper left')
pypl.title(r'Pendule non amorti : cas limite')
pypl.savefig('pendule-limite.pdf')
pypl.show()
pypl.clf()



# Exo 8

g = 10

def chute(y, t):
    [x, z, xp, zp] = y
    return [xp, zp, 0, -g]

def pommes(v0, alpha, tmax, n):
    t = np.linspace(0, tmax, n)
    values = odeint(chute, [0, 0, v0*np.cos(alpha), v0*np.sin(alpha)], t)
    valp = np.array([v for v in values if v[1] >= 0])
    pypl.plot(valp[:,0], valp[:,1])
```



```
v0=10.

for alpha in np.linspace(0, np.pi/2, 50):
    pommes(v0, alpha, 10, 1000)

x=np.linspace(0,v0**2/g,100)
pypl.plot(x,v0**2/2/g-g/v0**2*x*x/2,linewidth=4,color='black')

pypl.xlabel(r'$x(t)$',fontsize=18)
pypl.ylabel(r'$z(t)$',fontsize=18)
pypl.grid()
pypl.title('Chutes libres')
pypl.savefig('pommes-libres.pdf')
pypl.show()

pypl.clf()

k_sur_m=3
def chute_amortie(y, t):
    [x, z, xp, zp] = y
    return [xp, zp, -k_sur_m*xp, -g-k_sur_m*zp]

def pommes_amorties(v0, alpha, tmax, n):
    t = np.linspace(0, tmax, n)
    values = odeint(chute_amortie, [0, 0, v0*np.cos(alpha), v0*np.sin(alpha)], t
)
    valp = np.array([v for v in values if v[1] >= 0])
    pypl.plot(valp[:,0], valp[:,1])

v0=10
for alpha in np.linspace(0, np.pi/2, 50):
    pommes_amorties(v0, alpha, 10, 1000)

pypl.xlabel(r'$x(t)$',fontsize=18)
pypl.ylabel(r'$z(t)$',fontsize=18)
pypl.grid()
pypl.title('Chutes amorties')
pypl.savefig('pommes-amorties.pdf')
pypl.show()

pypl.clf()

# Exo 9

tmax=6
t=np.linspace(0,tmax,1000)

for alpha,beta in [(1,1), (10,1), (1,10)]:
    def cinet(c,_):
        ca,cb,_ = c
        return [-alpha*ca,alpha*ca-beta*cb,beta*cb]

    c=odeint(cinet,[1,0,0],t)

    pypl.plot(t,c[:,0],linewidth=2)
    pypl.plot(t,c[:,1],'--')
    pypl.plot(t,c[:,2],linewidth=4)

    pypl.xlabel(r'Temps',fontsize=18)
    pypl.ylabel(r'Concentrations',fontsize=18)
    pypl.axhline(color='black')

    pypl.grid()
    pypl.legend(['[A]','[B]','[C]'], loc = 'center right')

    pypl.savefig('cinetique-'+str(alpha)+'-'+str(beta)+'.pdf')
    pypl.show()
    pypl.clf()
```

```
# Exo 10

def instable(_, v):
    return np.array([v[1], 3*v[0]-2*v[1]-1])

def tableins(v, _):
    return np.array([v[1], 3*v[0]-2*v[1]-1])

n = 1000
pypl.grid()

tmax = 26
t = np.linspace(0, tmax, 1+n)
r = odeint(tableins, np.array([4./3, -3]), t)
pypl.plot(t, r[:, 0])

tmax = 37
t, r = np.linspace(0, tmax, 1+n), euler(instable, 0, np.array([4./3,-3]), tmax,
n)
pypl.plot(t, np.array(r)[:,0], '--', linewidth=2)

tmax = 36
t, r = np.linspace(0, tmax, 1+n), heun(instable, 0, np.array([4./3,-3]), tmax, n
)
pypl.plot(t, np.array(r)[:,0], linewidth=4)

tmax = 35
t, r = np.linspace(0, tmax, 1+n), RK4(instable, 0, np.array([4./3,-3]), tmax, n)
pypl.plot(t, np.array(r)[:,0], linewidth=3)
pypl.legend(['odeint','Euler','Heun','RK4'], loc='upper center')

pypl.xlabel(r'$t$', fontsize=18)
pypl.ylabel(r"$y(t)$", fontsize=18)
pypl.title(u'Une situation très instable')

pypl.savefig('instable.pdf')
pypl.show()
pypl.clf()

# Exo 11

mu = 1

def vdp(z, t):
    [x, y] = z
    return [y, mu*(1-x**2)*y-x]

b, n = 50, 1000
t = np.linspace(0, b, n)

z = odeint(vdp, [0, 3], t)
pypl.plot(z[:,0], z[:,1], '-.', linewidth=4)

z = odeint(vdp, [0.1,0], t)
pypl.plot(z[:,0], z[:,1], '--',linewidth=2, color='black')#, linewidth=2)

z = odeint(vdp, [3,0], t)
pypl.plot(z[:,0], z[:,1], linewidth=2)

pypl.axhline(color='black')
pypl.axvline(color='black')

pypl.xlabel(r'$x(t)$',fontsize=18)
pypl.ylabel(r"$x'(t)$",fontsize=18)
pypl.grid()
pypl.legend([r"$(x(0),x'(0))=(0,3)$", r"$(x(0),x'(0))=(1/10,0)$",\
                  r"$(x(0),x'(0))=(3,0)$"], loc = 'upper left')
```



```
pypl.savefig('vdp.pdf')
```