```python
# -*- coding: utf-8 -*-
"""
Éditeur de Spyder

Ce script temporaire est sauvegardé ici :
/home/willie/.spyder2/.temp.py
"""

import matplotlib.pyplot as plt
import math as math

#=============================================================================
# I - Comparaison entre les coordonnees cartesiennes et polaires CI simples
#=============================================================================
#=============================================================================
# Conditions initiales et constantes
#=============================================================================

G            = 1.
M            = 1.
m            = 1.
x0           = 1.
#y0           = 0.
#vx0          = 0.
vy0          = 1.
dt           = 10**-2
tfinal       = 100.

#=============================================================================
# Euler cartesien CI simples (on suppose y0=0 et vx0=0)
#=============================================================================
def euler_cartesien(x0,vy0,dt,tfinal):
    xpoint=0
    ypoint=vy0
    x=x0
    y=0
    energie=0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)
    if energie<0:
        print("Trajectoire elliptique")
    elif energie>0:
        print("Trajectoire hyperbolique")
    else:
        print("Trajectoire parabolique")

    t=0
    techantillon=0 # initialisation du compteur echantillons
#    les_xpoint=[xpoint]
#    les_ypoint=[ypoint]
    les_x=[x]
    les_y=[y]
    les_t=[t]
    les_energies=[energie]

    while t+dt <= tfinal:
        # accelerations selon x et y au point (x ; y)
        ax           = - x*G*M*m/(math.sqrt(x**2+y**2))**3
        ay           = - y*G*M*m/(math.sqrt(x**2+y**2))**3
        # calculs des positions x+dx et y+dy grace aux vitesses au point (x ; y)
        x            = x + xpoint*dt
        y            = y + ypoint*dt
        # mises à jours des vitesses grace aux accelerations au point (x ; y)
        xpoint       = xpoint + ax*dt
        ypoint       = ypoint + ay*dt
        # incrementation des temps
        t            = t+dt
        techantillon = techantillon+dt #compteur pour ne prelever que
```

```
                                    #des echantillons

        if techantillon >= float(tfinal/1000): #on preleve tous les milliemes
            energie = 0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)
            les_x.append(x)
            les_y.append(y)
            les_energies.append(energie)
            les_t.append(t)
            techantillon=0

    return les_x, les_y, les_t, les_energies

les_x, les_y, les_t, les_energies=euler_cartesien(x0,vy0,dt,tfinal)
ecart = 100*((max(les_energies)-min(les_energies))/min(les_energies))
print("Pourcentage de fluctuation de l'energie simulee (euler cartesiens) :",\
'%.4f' % ecart)

fig1=plt.figure(num=1)
plt.plot(les_x,les_y)
plt.grid()
plt.axis('equal')
titre = 'Trajectoire calculs euler cartesiens ; pas dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(0)+' ; vx0='+str(0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
plt.xlabel('$x$')
plt.ylabel('$y$')
#fig1.savefig('CartesienSimple.eps',dpi=fig1.dpi)
plt.show()
plt.close()

#=============================================================================
# Euler polaire CI simples (on suppose y0=0 et vx0=0)
#=============================================================================
def euler_polaire(x0,vy0,dt,tfinal):
    # initialisation de r, theta et des vitesses
    # correct uniquement pour y0=0 et vx0=0 et X0>=0
    r=float(x0)
    theta=0.
    rpoint=0.
    thetapoint=float(vy0)/float(r)
    energie=0.5*m*(rpoint**2+(r*thetapoint)**2)-G*M*m/r
    t=0
    techantillon=0
    les_x=[x0]
    les_y=[0]
#    les_rpoint=[rpoint]
#    les_thetapoint=[thetapoint]
    les_r=[r]
    les_theta=[theta]
    les_t=[t]
    les_energies=[energie]

    while t+dt <= tfinal:
        if r < 10**-5: # parfois r converge tres vite vers zero
            break      # si r petit on sort pour eviter les divisions par zero
        ar           = - G*M*m/(r**2)+r*(thetapoint)**2
        atheta       = - 2*rpoint*thetapoint/r
        r            = r + rpoint*dt
        theta        = theta + thetapoint*dt
        rpoint       = rpoint + ar*dt
        thetapoint   = thetapoint + atheta*dt
        t            = t+dt
        techantillon = techantillon+dt

        if techantillon>=float(tfinal/1000):
```

```
            energie      = 0.5*m*(rpoint**2+(r*thetapoint)**2)-G*M*m/r
            x            = r*math.cos(theta)
            y            = r*math.sin(theta)
            les_r.append(r)
            les_theta.append(theta)
            les_x.append(x)
            les_y.append(y)
            les_energies.append(energie)
            les_t.append(t)
            techantillon=0

    return les_x, les_y, les_t, les_energies, les_r, les_theta

les_x,les_y,les_t,les_energies,les_r, les_theta=euler_polaire(x0,vy0,dt,tfinal)
ecart = 100*((max(les_energies)-min(les_energies))/min(les_energies))
print("Pourcentage de fluctuation de l'energie simulee (euler polaires) :",\
 '%.4f' % ecart)

fig21=plt.figure(num=21)
plt.plot(les_x,les_y)
plt.grid()
plt.axis('equal')
titre = 'Trajectoire calculs euler polaires. \n dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(0)+' ; vx0='+str(0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
plt.xlabel('$x$')
plt.ylabel('$y$')
#fig21.savefig('PolaireSimple1.eps',dpi=fig21.dpi)
plt.show()
plt.close()

fig22=plt.figure(num=22)
plt.polar(les_theta,les_r)
titre = 'Trajectoire calculs euler polaires ; pas dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(0)+' ; vx0='+str(0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
#fig22.savefig('PolaireSimple2.eps',dpi=fig22.dpi)
plt.show()
plt.close()

#=============================================================================
# II - Comparaison entre methodes Euler et Leap-frog CI quelconques
#=============================================================================
#=============================================================================
# Conditions initiales et constantes
#=============================================================================

G            = 1.
M            = 1.
m            = 1.
x0           = 1.1
y0           = 0.5
vx0          = 0.2
vy0          = 0.7
dt           = 10**-2
tfinal       = 100.

#=============================================================================
# Leapfrog cartesien CI quelconques
#=============================================================================
def euler_leapfrog_CI_quelconques(position,vitesse,dt,tfinal):

    x,y=position # x0 et y0
```

```
    if x==0 and y==0:
            print("Ne pas prendre x et y initiaux tous deux nuls !" )
            return
    else:

        xpoint,ypoint=vitesse #vx0 et vy0

        energie=0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)

        if energie<0:
            print("trajectoire elliptique")
        elif energie>0:
            print("trajectoire hyperbolique")
        else:
            print("trajectoire parabolique")

        #Leap frog : initialisation particuliere de la vitesse avec un demi pas
        xpoint = xpoint - 0.5*x*G*M*m/(math.sqrt(x**2+y**2))**3*dt #vx1/2 = vx0 +
0.5*ax0*dt
        ypoint = ypoint - 0.5*y*G*M*m/(math.sqrt(x**2+y**2))**3*dt #vy1/2 = vy0 +
0.5*ay0*dt

        t=0
        techantillon=0 #compteur pour ne prelever que des echantillons

    # definition des listes pour stocker les echantillons
    #    les_xpoint=[xpoint]
    #    les_ypoint=[ypoint]
        les_x=[x]
        les_y=[y]
        les_t=[t]
        les_energies=[energie]

        while t+dt <= tfinal:
            x            = x + xpoint*dt #x_n+1 = x_n + vx_n+1/2 * dt
            y            = y + ypoint*dt #y_n+1 = y_n + vy_n+1/2 * dt

            #vx_n+3/2 = vx_n+1/2 + ax_n+1 * dt
            #vy_n+3/2 = vy_n+1/2 + ay_n+1 * dt
            xpoint       = xpoint - x*G*M*m/(math.sqrt(x**2+y**2))**3*dt
            ypoint       = ypoint - y*G*M*m/(math.sqrt(x**2+y**2))**3*dt
            t            = t+dt
            techantillon = techantillon+dt

            if techantillon >= float(tfinal/1000): # stockage de mille points
                energie      = 0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)
                les_x.append(x)
                les_y.append(y)
                les_energies.append(energie)
                les_t.append(t)
                techantillon=0

        return les_x, les_y, les_t, les_energies

les_x, les_y, les_t, les_energies=euler_leapfrog_CI_quelconques((x0,y0),-
(vx0,vy0),dt,tfinal)
ecart = 100*((max(les_energies)-min(les_energies))/min(les_energies))
print("Pourcentage de fluctuation de l'energie simulee (leapfrog cartesiens) :",\
 '%.4f' % ecart)

fig3=plt.figure(num=3)
plt.plot(les_x,les_y)
plt.grid()
plt.axis('equal')
titre = 'Trajectoire calculs leapfrog cartesiens ; pas dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(y0)+' ; vx0='+str(vx0)\
```

```
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
plt.xlabel('$x$')
plt.ylabel('$y$')
#fig3.savefig('LeapFrogComplexe.eps',dpi=fig3.dpi)
plt.show()
plt.close()

#=============================================================================
# Euler cartesien CI quelconques
#=============================================================================
def euler_cartesien_CI_quelconques(position,vitesse,dt,tfinal):

    x,y=position

    if x==0 and y==0:
            print("Ne pas prendre x et y initiaux tous deux nuls !")
            return

    else:

        xpoint,ypoint=vitesse
        energie=0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)

        if energie<0:
            print("trajectoire elliptique")
        elif energie>0:
            print("trajectoire hyperbolique")
        else:
            print("trajectoire parabolique")

        t=0
        techantillon=0
    #    les_xpoint=[xpoint]
    #    les_ypoint=[ypoint]
        les_x=[x]
        les_y=[y]
        les_t=[t]
        les_energies=[energie]

        while t+dt <= tfinal:
            ax           = - x*G*M*m/(math.sqrt(x**2+y**2))**3
            ay           = - y*G*M*m/(math.sqrt(x**2+y**2))**3
            x            = x + xpoint*dt
            y            = y + ypoint*dt
            xpoint       = xpoint + ax*dt
            ypoint       = ypoint + ay*dt
            t            = t+dt
            techantillon = techantillon+dt

            if techantillon >= float(tfinal/1000):
                energie      = 0.5*m*(xpoint**2+ypoint**2)-G*M*m/math.sqrt(x**2+y**2)
                les_x.append(x)
                les_y.append(y)
                les_energies.append(energie)
                les_t.append(t)
                techantillon=0

        return les_x, les_y, les_t, les_energies

les_x, les_y, les_t, les_energies=euler_cartesien_CI_quelconques((x0,y0),-
(vx0,vy0),dt,tfinal)
ecart = 100*((max(les_energies)-min(les_energies))/min(les_energies))
print("Pourcentage de fluctuation de l'energie simulee (euler cartesiens) :", '%.4f' %
ecart)
```

```
fig5=plt.figure(num=5)
plt.plot(les_x,les_y)
plt.grid()
plt.axis('equal')
titre='Trajectoire calculs euler cartesiens ;pas dt='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(y0)+' ; vx0='+str(vx0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
plt.xlabel('$x$')
plt.ylabel('$y$')
#fig5.savefig('CartesienComplexe.eps',dpi=fig5.dpi)
plt.show()
plt.close()

#=============================================================================
# Euler polaire CI quelconques
#=============================================================================
def euler_polaire_CI_quelconques(position,vitesse,dt,tfinal):

    x,y=position

    if x!=0:
        if x >0:
            theta=math.atan(y/x)
        else:
            theta=math.pi+math.atan(y/x)
    else:
        if y > 0:
            theta=math.pi/2
        elif y <0:
            theta=-math.pi/2
        else:
            print("Ne pas prendre x et y initiaux nuls !")
            return

    r=math.sqrt(x**2+y**2)
    xpoint,ypoint=vitesse
    rpoint=xpoint*x/math.sqrt(x**2+y**2)+ypoint*y/math.sqrt(x**2+y**2)
    thetapoint=1/r*(-xpoint*y/math.sqrt(x**2+y**2)+ypoint*x/math.sqrt(x**2+y**2))
    energie=0.5*m*(rpoint**2+(r*thetapoint)**2)-G*M*m/r

    if energie<0:
        print("trajectoire elliptique")
    elif energie>0:
        print("trajectoire hyperbolique")
    else:
        print("trajectoire parabolique")

    t=0
    techantillon=0
    les_x=[x]
    les_y=[y]
#    les_rpoint=[rpoint]
#    les_thetapoint=[thetapoint]
    les_r=[r]
    les_theta=[theta]
    les_t=[t]
    les_energies=[energie]

    while t+dt <= tfinal:
        if r < 10**-5: # parfois r converge tres vite vers zero
            break      # si r petit on sort pour eviter les divisions par zero
        ar          = - G*M*m/(r**2)+r*(thetapoint)**2
        atheta      = - 2*rpoint*thetapoint/r
        r           = r + rpoint*dt
        theta       = theta + thetapoint*dt
```

```
        rpoint       = rpoint + ar*dt
        thetapoint   = thetapoint + atheta*dt
        t            = t+dt
        techantillon = techantillon+dt

        if techantillon>=float(tfinal/1000):
            energie      = 0.5*m*(rpoint**2+(r*thetapoint)**2)-G*M*m/r
            x            = r*math.cos(theta)
            y            = r*math.sin(theta)
            les_r.append(r)
            les_theta.append(theta)
            les_x.append(x)
            les_y.append(y)
            les_energies.append(energie)
            les_t.append(t)
            techantillon=0

    return les_x, les_y, les_t, les_energies, les_r, les_theta

les_x,les_y,les_t,les_energies,les_r,les_theta=euler_polaire_CI_quelconques((x0,y0),-
(vx0,vy0),dt,tfinal)
ecart = 100*((max(les_energies)-min(les_energies))/min(les_energies))
print("Pourcentage de fluctuation de l'energie simulee (euler polaires) :",\
 '%.4f' % ecart)

fig6=plt.figure(num=6)
plt.plot(les_x,les_y)
plt.grid()
plt.axis('equal')
titre='Trajectoire calculs euler polaires ; pas dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(y0)+' ; vx0='+str(vx0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
plt.xlabel('$x$')
plt.ylabel('$y$')
#fig6.savefig('PolaireComplexe.eps',dpi=fig6.dpi)
plt.show()
plt.close()

fig7=plt.figure(num=7)
plt.polar(les_theta,les_r)
titre='Trajectoire calculs euler polaires ; pas dt ='\
+str(dt)+'\n x0='+str(x0)+' ; y0='+str(y0)+' ; vx0='+str(vx0)\
+' ; vy0='+str(vy0)+'\n ecart en energie :'+'%.2f' % ecart+' %'
plt.title(titre)
#fig7.savefig('PolaireComplexe2.eps',dpi=fig7.dpi)
plt.show()
plt.close()
```