Python et Physique
Physique - Chimie

1- Caractéristique d'une CTN

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100
from scipy import interpolate

T = np.array([2, 6, 10, 15, 20, 25, 30, 35, 40, 45, 50.1, 55, 60, 65, 70, 75, 80, 85])
R = np.array([25378, 21487, 18301, 14990, 12402, 10295, 8615, 7226, 6097, 5121, 4306, 3632, 3070,␣
↪2609, 2221, 1903, 1630, 1404])

#Interplotation
f = interpolate.interp1d(T, R, kind='cubic')
Tnew = np.linspace(T.min(), T.max(), 100)
Rnew = f(Tnew)

plt.plot(Tnew, Rnew)
plt.plot(T,R,'x')
plt.title('$R=f(T)$')
plt.xlabel('$T(°C)$')
plt.xlim(0,100)
plt.ylabel('$R(\Omega)$')
plt.ylim(0,30000)
plt.grid()
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC1.py.

Une fois cela fait, appelez le professeur pour vérification.

2- Mouvement d'un point - Positions

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([0.003, 0.141, 0.275, 0.410, 0.554, 0.686, 0.820, 0.958, 1.089, 1.227, 1.359, 1.490,␣
↪1.599, 1.705, 1.801])
y = np.array([0.746, 0.990, 1.175, 1.336, 1.432, 1.505, 1.528, 1.505, 1.454, 1.355, 1.207, 1.018,␣
↪0.797, 0.544, 0.266])

plt.plot(x, y, ".")
plt.xlabel("x (m)")
plt.ylabel("y (m)")
plt.grid(True)
plt.title("Trajectoire d'un ballon")
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC2.py.

Une fois cela fait, appelez le professeur pour vérification.

3- Mouvement d'un point - Vecteur vitesse

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt

# Données
dt = 0.0667
x = np.array([0.003, 0.141, 0.275, 0.410, 0.554, 0.686, 0.820, 0.958, 1.089, 1.227, 1.359, 1.490,␣
↪1.599, 1.705, 1.801])
y = np.array([0.746, 0.990, 1.175, 1.336, 1.432, 1.505, 1.528, 1.505, 1.454, 1.355, 1.207, 1.018,␣
↪0.797, 0.544, 0.266])

# Calculs des vecteurs vitesses
N = len(x)          # Nombre de points de mesures
vx = np.zeros(N)    # Initialisation d'un tableau vide
vy = np.zeros(N)    # Idem
for i in range(1, N-1):
    vx[i]=(x[i+1]-x[i-1])/(2*dt)
    vy[i]=(y[i+1]-y[i-1])/(2*dt)

# Calcul des vitesses
v = np.sqrt(vx**2+vy**2)

plt.plot(x, y, '.')
plt.xlabel("x (m)")
plt.xlim(0, 2)
plt.ylabel("y (m)")
plt.ylim(0, 2)
plt.title("Trajectoire d'un ballon")
for i in range(1,N-1):
    plt.arrow(x[i], y[i], vx[i]/10, vy[i]/10, head_width=0.025, color='r')
    plt.annotate(i, (x[i]-0.05,y[i]-0.15))
    print('Point', i, '-> v=', round(v[i],2), " m/s")
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC3.py.

Une fois cela fait, appelez le professeur pour vérification.

Quelques remarques:

- La fonction matplotlib.pyplot.arrow() permet de tracer des flèches pour illustrer les graphiques.
- La fonction matplotlib.pyplot.quiver() a l'avantage d'être itérative. Elle permet de tracer un champ de vecteurs.

Modifier le programme précédant de la façon suivante.

```
import numpy as np
import matplotlib.pyplot as plt

# Données
dt = 0.0667
x = np.array([0.003, 0.141, 0.275, 0.410, 0.554, 0.686, 0.820, 0.958, 1.089, 1.227, 1.359, 1.490,␣
↪1.599, 1.705, 1.801])
y = np.array([0.746, 0.990, 1.175, 1.336, 1.432, 1.505, 1.528, 1.505, 1.454, 1.355, 1.207, 1.018,␣
↪0.797, 0.544, 0.266])

# Calculs des vecteurs vitesses
N = len(x)          # Nombre de points de mesures
vx = np.zeros(N)    # Initialisation d'un tableau vide
vy = np.zeros(N)    # Idem
for i in range(1, N-1):
    vx[i]=(x[i+1]-x[i-1])/(2*dt)
    vy[i]=(y[i+1]-y[i-1])/(2*dt)

# Calcul des vitesses
v = np.sqrt(vx**2+vy**2)

plt.plot(x, y, ".")
plt.xlabel("x (m)")
plt.xlim(0, 2)
plt.ylabel("y (m)")
plt.ylim(0, 2)
plt.title("Trajectoire d'un ballon")
plt.quiver(x, y, vx, vy, angles='xy', scale_units='xy', scale=10, color='red', width=0.005)
for i in range(1, N-1):
    plt.annotate(i, (x[i]-0.05,y[i]-0.15))
    print('Point', i, '-> v=', round(v[i],2), " m/s")
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC4.py.

Une fois cela fait, appelez le professeur pour vérification.

4- Mouvement d'un point - Vecteur variation de vitesse

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100

dt = 0.066
x = np.array([0.003, 0.141, 0.275, 0.410, 0.554, 0.686, 0.820, 0.958, 1.089, 1.227, 1.359, 1.490,␣
↪1.599, 1.705, 1.801])
y = np.array([0.746, 0.990, 1.175, 1.336, 1.432, 1.505, 1.528, 1.505, 1.454, 1.355, 1.207, 1.018,␣
↪0.797, 0.544, 0.266])
N = x.size

vx = np.zeros(N)
vy = np.zeros(N)
for i in range(1, N-1):
    vx[i] = (x[i+1]-x[i-1])/(2*dt)
    vy[i] = (y[i+1]-y[i-1])/(2*dt)
```

```
#print(vx.round(2))
#print(vy.round(2))

dvx = np.zeros(N)
dvy = np.zeros(N)
for i in range(2, N-2):
    dvx[i] = vx[i+1]-vx[i-1]
    dvy[i] = vy[i+1]-vy[i-1]

#print(dvx.round(2))
#print(dvy.round(2))

plt.plot(x, y, ".")
plt.quiver(x, y, vx, vy, angles='xy', scale_units='xy', scale=10, color='red', width=0.005)
plt.quiver(x, y, dvx, dvy, angles='xy', scale_units='xy', scale=3, color='blue', width=0.005)
plt.xlabel("x (m)")
plt.xlim(0, 2)
plt.ylabel("y (m)")
plt.ylim(0, 2)
plt.title("Trajectoire d'un ballon")
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC5.py.

Remarque:

- La fonction quiver() permet de tracer un vecteur ou un champ de vecteur.

Une fois cela fait, appelez le professeur pour vérification.

5- Mouvement d'un point - Vecteur accélération

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 100
m=2.5
dt = 0.066
x = np.array([0.003, 0.141, 0.275, 0.410, 0.554, 0.686, 0.820, 0.958, 1.089, 1.227, 1.359, 1.490, 1.599, 1.705, 1.801])
y = np.array([0.746, 0.990, 1.175, 1.336, 1.432, 1.505, 1.528, 1.505, 1.454, 1.355, 1.207, 1.018, 0.797, 0.544, 0.266])
N = x.size
vx = np.zeros(N)
vy = np.zeros(N)
for i in range(1, N-1):
    vx[i] = (x[i+1]-x[i-1])/(2*dt)
    vy[i] = (y[i+1]-y[i-1])/(2*dt)
#print(vx.round(2))
#print(vy.round(2))
dvx = np.zeros(N)
dvy = np.zeros(N)
for i in range(2, N-2):
    dvx[i] = vx[i+1]-vx[i-1]
    dvy[i] = vy[i+1]-vy[i-1]
#print(dvx.round(2))
#print(dvy.round(2))
plt.plot(x, y, "x")
plt.quiver(x, y, vx, vy, angles='xy', scale_units='xy', scale=10, color='red', width=0.005)
plt.quiver(x, y, m*dvx, m*dvy, angles='xy', scale_units='xy', scale=3, color='blue', width=0.005)
plt.xlabel("x (m)")
plt.xlim(0, 2)
plt.ylabel("y (m)")
plt.ylim(0, 2)
plt.title("Trajectoire d'un ballon")
plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC6.py.

Une fois cela fait, appelez le professeur pour vérification.

6- Bilan énergétique d'un mouvement rectiligne de chute libre

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import matplotlib.pyplot as plt
from math import sqrt
plt.rcParams['figure.dpi'] = 70

fig, ax = plt.subplots()

m, g, z0, t0 = 0.5, 9.81, 50.0, 0.0

v = [-g*t0]
z = [0.5*g*t0**2+z0]
dt=sqrt(2*z0/g)
Ep = [-0.5*m*(g*t0)**2+m*g*z0]
Ec = [0.5*m*(g*t0)**2]
Em=[-0.5*m*(g*t0)**2+m*g*z0 + 0.5*m*(g*t0)**2]
T = [t0]

for t in range(0, int(dt*100), 1):
    T.append(t/100)
    z.append(-0.5*g*(t/100)**2+z0)
    Ep.append(-0.5*m*(g*t/100)**2+m*g*z0)
    Ec.append(0.5*m*(g*t/100)**2)
    Em.append(-0.5*m*(g*t/100)**2+m*g*z0 + 0.5*m*(g*t/100)**2)

plt.plot(T, Ep, label="Ep")
plt.plot(T, Ec, label="Ec")
plt.plot(T, Em, label="Ec")
plt.xlabel("Durée de chute(s)")
plt.ylabel("Energies(J)")
plt.title("Bilan énergétique - Chute libre d'une balle")
plt.legend()
plt.grid()

fig.savefig("Bilan énergétique - Chute libre d'une balle")

plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC7.py.

Une fois cela fait, appelez le professeur pour vérification.

7- Evolution des énergies d'un système en mouvement dans un champ uniforme

En utilisant Spyder recopier et exécuter le programme ci-dessous.

```
import matplotlib.pyplot as plt
from math import sqrt, sin, cos
import math
plt.rcParams['figure.dpi'] = 70

fig, ax = plt.subplots()

m, g, z0, t0, v0, alpha = 0.5, 9.81, 0.0, 0.0, 4.52, 67.0
alpha = math.radians(alpha)

c = cos(alpha)
s = sin(alpha)
vx = [v0*c]
vz = [-g*t0*v0*s]
v = [sqrt((v0*c)**2+(-g*t0+v0*s)**2)]
x = [v0*c]
z = [-0.5*g*t0**2+v0*s*t0+z0]
dt = 2*v0*s/g
Ep = [-0.5*m*(g*t0)**2+m*g*v0*s*t0+m*g*z0]
Ec = [0.5*m*((v0*c)**2+(-g*t0+v0*s)**2)]
Em = [-0.5*m*(g*t0)**2+m*g*v0*s*t0+m*g*z0 + 0.5*m*((v0*c)**2+(-g*t0+v0*s)**2)]
T = [t0]

for t in range(0, int(dt*100), 1):
    T.append(t/100)
    z.append(-0.5*g*(t/100)**2 + v0*s*(t/100)+z0)
    Ep.append(-0.5*m*(g*t/100)**2 + m*g*v0*s*t/100 + m*g*z0)
    Ec.append(0.5*m*((v0*c)**2 + (-g*t/100+v0*s)**2))
    Em.append(-0.5*m*(g*t/100)**2+m*g*v0*s*t/100+m*g*z0 + 0.5*m*((v0*c)**2+(-g*t/100+v0*s)**2))

print(Ep[5])
print(Ec[5])
print(Em[5])

plt.plot(T, Ep, label="Ep")
plt.plot(T, Ec, label="Ec")
plt.plot(T, Em, label="Em")
plt.xlabel("Durée (s)")
plt.ylabel("Energies(J)")
plt.title("Bilan énergétique - Lancer d'une balle")
plt.legend()
plt.grid()

fig.savefig("Bilan énergétique - Lancer d'une balle")

plt.show()
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC8.py.

Une fois cela fait, appelez le professeur pour vérification.

8- Les lois de Kepler

En utilisant Spyder recopier et exécuter le programme ci-dessous.

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC9.py.

Une fois cela fait, appelez le professeur pour vérification.

9- Représentation d'une onde

En utilisant Spyder recopier et exécuter le programme ci-dessous.

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC10.py.

Une fois cela fait, appelez le professeur pour vérification.

10- Simuler la propagation d'une onde

En utilisant Spyder recopier et exécuter le programme ci-dessous.

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC11.py.

Une fois cela fait, appelez le professeur pour vérification.

11- Interférence de deux ondes lumineuses

En utilisant Spyder recopier et exécuter le programme ci-dessous.

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom PC12.py.

Une fois cela fait, appelez le professeur pour vérification.