Python et Physique
Calculs Scientifiques

1- Interpolation de courbes

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

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

x=np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y=np.array([1., 0.720, 0.511, 0.371, 0.260, 0.190, 0.133, 0.099, 0.069, 0.048])

# Interpolation
f = interpolate.interp1d(x, y, kind='cubic')  # ou kind = 'linear'

xnew = np.linspace(0, 9, 50)        # Nouvelle abscisse pour f(x)
ynew = f(xnew)                      # Calcul des ordonnées de f(x)

plt.plot(xnew, ynew, '-')           # Tracé de la fonction
plt.plot(x, y, 'x')                 # Tracé des points
plt.show()                          # Affichage
```

Sauvegarder ce programme dans un dossier nommé SciPython sous le nom prg8.py.

Remarque:

- La fonction scipy.interpolate.interp1d() retourne une fonction (mathématique) à une dimension interpolée à partir d'une série de points.

Une fois cela fait, appelez le professeur pour vérification.

2- Régression linéaire

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

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

x = np.array([0, 1.01, 2.02, 2.99, 3.98])
y = np.array([10.02, 7.96, 6.03, 4.04, 2.01])


a, b, rho ,_ ,_ = linregress(x, y)    # Régression linéaire
print("a = ", a)                      # Affichage de coefficient directeur
print("b = ", b)                      # Affichage de l'ordonnée à l'origine
print("rho = ", rho)                  # Affichage du coefficient de corrélation

xnew = np.linspace(0, 4, 50)          # Nouvelle abscisse pour la modélisation
ynew = a*xnew + b                     # Ordonnées de la fonction affine
```

```
plt.plot(xnew, ynew, '-')              # Tracé de la droite
plt.plot(x, y, 'x')                    # Tracé des points et de la fonction affine
plt.title('Régression linéaire')       # Titre
plt.xlabel('x')                        # Etiquette en abscisse
plt.xlim(-1,5)                         # Echelle en abscisse
plt.ylabel('y')                        # Etiquette en ordonnée
plt.ylim(0, 12)                        # Echelle en ordonnée
plt.show()                             # Affichage
```

Sauvegarder ce programme dans le dossier nommé SciPython sous le nom prg9.py.

Remarque:

- La fonction scipy.stats.linregress() retourne les paramètres de la régression linéaire d'une série de points.

Une fois cela fait, appelez le professeur pour vérification.

3- Modélisation à partir d'un polynome

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

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

# Donnée expérimentale
T = [0.0, 0.04, 0.08, 0.12, 0.16, 0.2, 0.24, 0.28, 0.32, 0.36, 0.4, 0.44, 0.48, 0.52, 0.56, 0.6,␣
↪0.64, 0.68, 0.72, 0.76, 0.8, 0.84, 0.88, 0.92]
X = [-0.953328037081172, -0.879995111151852, -0.799995555592592, -0.716662685218364, -0.
↪636663129659105, -0.559996888914815, -0.479997333355555, -0.393331148166358, -0.313331592607099,
↪ -0.233332037047839, -0.149999166673611, -0.066666296299383, 0.013333259259877, 0.
↪096666129634105, 0.179999000008333, 0.259998555567592, 0.343331425941821, 0.426664296316049, 0.
↪506663851875308, 0.586663407434568, 0.663329648178858, 0.743329203738117, 0.819995444482407, 0.
↪893328370411728]
Y = [-0.046666407409568, 0.069999611114352, 0.166665740748457, 0.253331925937654, 0.
↪326664851866975, 0.389997833351389, 0.433330925945988, 0.469997388910648, 0.486663962985494, 0.
↪493330592615432, 0.489997277800463, 0.469997388910648, 0.433330925945988, 0.38666451853642, 0.
↪323331537052006, 0.249998611122685, 0.156665796303549, 0.053333037039506, -0.063332981484414, -
↪0.189998944453241, -0.333331481496913, -0.486663962985494, -0.65332970373395, -0.
↪789995611147685]

a, b, c = np.polyfit(X, Y, 2)  # Modélisation par un polynome de degré 2

print("a = ", a)               # Affichage
print("b = ", b)
print("c = ", c)

X = np.array(X)                # Création d'un tableau Numpy pour X
Y_modele = a*X**2+b*X+c        # Création d'un tableau Numpy pour Y du modèle

plt.plot(X,Y,'b+', label = 'trajectoire')       # Courbe des mesures
plt.plot(X, Y_modele, 'r-', label = "modèle")   # Courbe du modèle
plt.xlabel("x")                                 # Etiquette en abscisse
plt.ylabel("y")                                 # Etiquette en ordonnée
plt.title("Trajectoire et modèle associé")      # Titre
plt.legend()                                    # Affichage légend
plt.show()                                      # Affichage fenêtre
```

Sauvegarder ce programme dans le dossier nommé SciPython sous le nom prg10.py.

Une fois cela fait, appelez le professeur pour vérification.

Quel type de courbe obtient-on?

4- Modélisation à partir d'une fonction quelconque

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

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

x=np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y=np.array([0., 3.935, 6.321, 7.769, 8.647, 9.179, 9.502, 9.698, 9.817, 9.889])


def fct(x, A, tau):                    # Définition de la fonction
    return A*(1-np.exp(-x/tau))        # Expression du modèle

(A, tau), pcov = curve_fit(fct, x, y)  # Détermination des paramètres du modèle
print("A = ", A)                       # Affichage de A
print("tau =", tau)                    # Affichage de tau

xnew = np.linspace(0, 10, 50)
ynew = fct(xnew, A, tau)
plt.plot(xnew, ynew, '-')
plt.plot(x, y, 'x')
plt.show()
```

Remarque:

- La fonction scipy.optimize.curve_fit() retourne les paramètres de modélisation à partir d'une fonction quelconque.

Sauvegarder ce programme dans le dossier nommé SciPython sous le nom prg11.py.

Une fois cela fait, appelez le professeur pour vérification.