ON 3 - Régression linéaire

Ce qu'il faut savoir et savoir faire

→ Utiliser un logiciel de régression linéaire afin d'obtenir les valeurs des paramètres du modèle.
→ Analyser les résultats obtenus à l'aide d'une procédure de validation : analyse graphique intégrant les barres d'incertitude ou analyse des écarts normalisés.

Pour commencer, on entre les valeurs des entrées $\boldsymbol{x}$ et $\boldsymbol{y}$ sous la forme de deux **tableaux de valeurs**.

I. Faire une régression linéaire

1. Algorithme Python

Dans la bibliothèque `numpy` se trouve la fonction **`polyfit`**, qui permet de trouver le polynôme qui modélise le mieux la courbe expérimentale $y(x)$. **Pour une régression linéaire, il s'agit de chercher un polynôme de degré 1.**

La commande `[a,b]=np.polyfit(x,y,1)` renvoie les coefficients a et b qui modélisent le mieux la série de données selon une relation affine $y = ax + b$.

2. Exemple

On mesure une distance d parcourue en fonction d'un temps t de parcours.

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

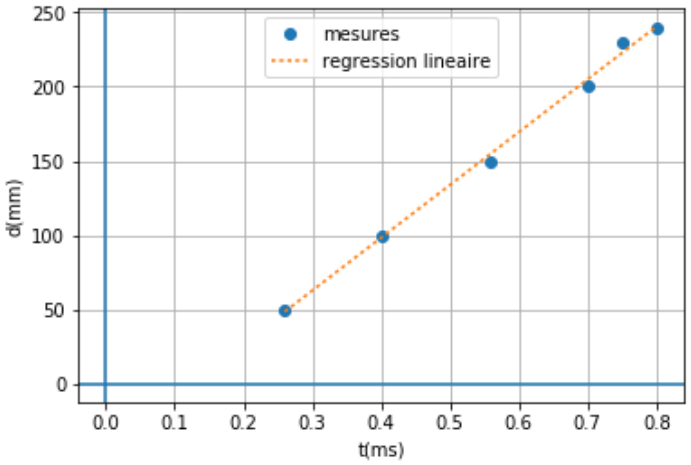
#les mesures
t=np.array([0.26,0.40,0.56,0.70, 0.75,0.80]) #durée en secondes
d=np.array([50,100,150,200,230,240])  #distance en mm
plt.plot(t,d,'o',label='mesures')

#regression linéaire
[a,b]=np.polyfit(t,d,1)
print("a = {:.5f}".format(a))# 5 chiffres après la virgule
print("b = {:.5f}".format(b))

#tracé du modèle
d_model=a*t+b
plt.plot(t,d_model,':',label='regression lineaire')
plt.legend()
plt.grid()
plt.axhline()
plt.axvline()
plt.xlabel('t(ms)')
plt.ylabel('d(mm)')
plt.show()
```

Résultats obtenus :

```
a = 354.88136
b = -43.57306
```



II. Validation du modèle

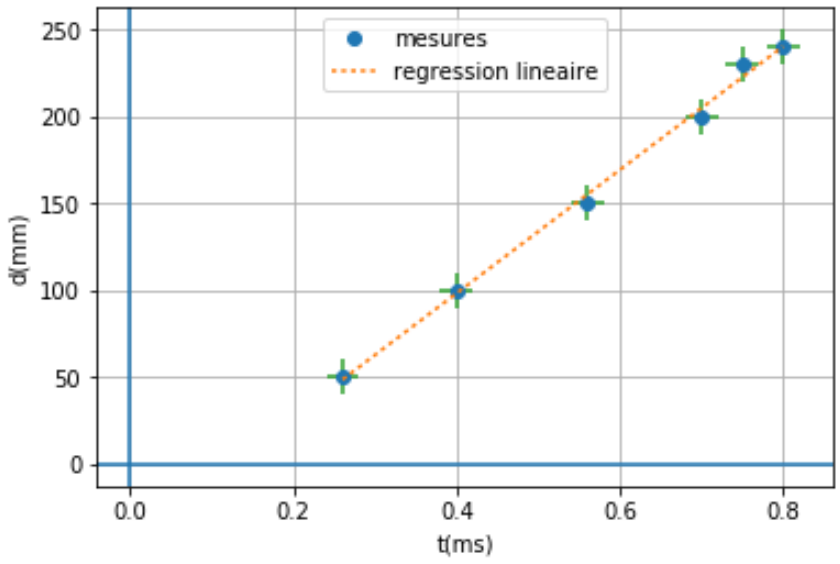
Un premier contrôle visuel permet de valider le modèle.

1. Barres d'incertitudes

La commande `plt.errorbar(x,y,xerr = 'valeur de Δx',yerr = 'valeur de Δy')` permet de tracer les barres d'incertitudes.

Dans l'exemple étudié, on suppose les précisions constantes. On rajoute à l'algorithme précédent :

```
#barres d'incertitudes
delta_t=0.02  # précision de t (valeur constante)
delta_d=10    # précision de d (valeur constante)
plt.errorbar(t, d, xerr = delta_t, yerr = delta_d,linestyle='none')
plt.show()
```



2. Estimation des incertitudes par la méthode de Monte Carlo

On veut déterminer les incertitudes-types sur les coefficients a et b du modèle $y = ax + b$.
On considère une série de m mesures de couples (x_i, y_i), on note Δx_i *et* Δy_i les demi-largeurs des intervalles auxquels appartiennent respectivement les valeurs x_i *et* y_i. Pour estimer l'incertitude-type du coefficient directeur a et de l'ordonnée à l'origine b, on utilise alors la méthode de Monte-Carlo : on simule informatiquement N régressions linéaires.

1. **Entrer les grandeurs expérimentales (x_i, y_i) dans de deux tableaux.**
2. **Lister les incertitudes-type ou les précisions de ces grandeurs** afin d'associer à chaque mesure, un intervalle au sein duquel on peut raisonnablement penser que celle-ci appartient.
3. **Initialiser deux listes vides qui contiendront les valeurs de a et b.**
4. **Fixer un nombre N de simulations à réaliser**. Pour chacune :
   → **Réaliser un tirage aléatoire de m valeurs** dans les intervalles $[x_i - \Delta x_i, x_i + \Delta x_i]$ et $[y_i - \Delta y_i, y_i + \Delta y_i]$.
   → **Réaliser une régression linéaire** sur cet ensemble de valeurs (m couples (x, y)) avec `polyfit`.
   → **Stocker** dans les listes la pente et l'ordonnée à l'origine de cette régression.
5. **Calculer les valeurs moyennes de a et b.**
6. **Calculer les écarts-types** des deux listes pour **obtenir les incertitudes-types de a et b.**

```python
# Entrée des données
t=np.array([0.26,0.40,0.56,0.70, 0.75,0.80])
d=np.array([50,100,150,200,230,240])
delta_t = 0.02
delta_d = 10

# Nombre de simulations
N = 10000

# Listes des valeurs du coefficient directeur et de l'ordonnée à l'origine
A = []
B = []

# Réalisation de la simulation
for k in range(N):
    t_sim = t+np.random.uniform(-delta_t,delta_t,len(t))
    d_sim = d+np.random.uniform(-delta_d,+delta_d,len(d))
    a,b = np.polyfit(t_sim,d_sim,1)  #régression linéaire
    A.append(a)
    B.append(b)
# Calcul des moyennes
a_moy=np.mean(A)
b_moy=np.mean(B)
# Calcul des incertitudes-type
u_a=np.std(A,ddof=1)
u_b=np.std(B,ddof=1)

# Affichage des résultats avec 5 chiffres après la virgule
print("u(a) = {:.5f} ".format(u_a))
print("u(b) = {:.5f} ".format(u_b))
```

Résultats obtenus : `a = 354.35173, u(a) = 14.76266, b = -43.27990, u(b) = 8.98458`