Python pour les incertitudes en TP

I La bibliothèque numpy

Le module `numpy` permet de créer et de manipuler facilement des tableaux appelés `arrays`. En travaux pratiques, on utilisera ce module pour stocker et traiter des données expérimentales, sans avoir besoin d'écrire systématiquement des boucles (ce qui serait le cas si on manipulait des listes). L'utilisation des tableaux `numpy` et des fonctions associées est **l'analogue en Python de l'utilisation d'un tableur.**

On pourra se référer à **l'Aide-mémoire NUMPY pour les activités expérimentales en Physique-Chimie.**

II Calcul d'une moyenne et d'un écart-type avec Python

☞ En langage Python on peut :

— calculer **la moyenne** d'un tableau de valeurs avec la fonction `np.mean` (de l'anglais *mean* pour moyenne) ;

— calculer **l'écart-type** d'un tableau de valeurs avec la fonction `np.std` (*std* de l'anglais *standard-deviation* pour écart-type) .

Exemple 1

En TP, un·e élève cherche à mesurer la période des oscillations d'un pendule élastique (une masse suspendue à un ressort vertical). L'élève mesure 10 fois la durée T_5 de 5 oscillations du pendule, à l'aide d'un chronomètre manuel. Les valeurs obtenues sont consignées dans le tableau suivant :

T_5 (s)	3,78	3,84	3,91	3,84	3,91	3,81	3,81	3,85	3,82	3,82

✍ Calculer la moyenne de la série de mesures, son écart-type, et l'incertitude-type sur la moyenne.

→ **Réponse :** on peut utiliser le programme suivant.

```
import numpy as np

#tableau des valeurs mesurees
T5_data = np.array([3.78, 3.84, 3.91, 3.84, 3.91, 3.81, 3.81, 3.85, 3.82, 3.82])

#moyenne des valeurs mesurees
moyenne = np.mean(T5_data)

#écart-type des valeurs mesurees
ecart_type = np.std(T5_data, ddof = 1)

#écart-type de la moyenne
u_T5 = ecart_type/np.sqrt(10)
```

III Tracé d'un histogramme

☞ En langage Python on peut :

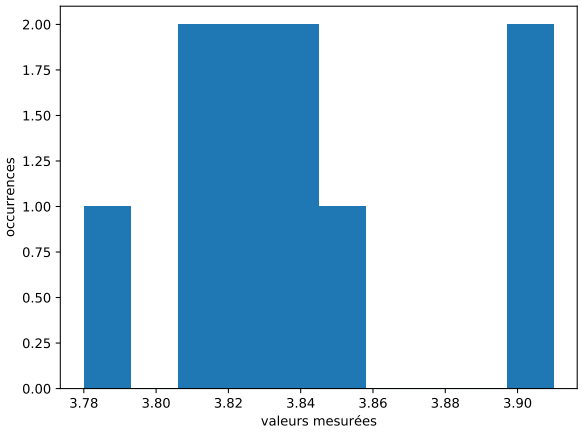
— tracer **l'histogramme** d'un tableau de valeurs avec la fonction `plt.hist` .

Exemple 2

On reprend la liste des valeurs mesurées dans **l'exemple 1** ci-dessus. ✍ Tracer l'histogramme des valeurs mesurées.

→ **Réponse :** on peut utiliser le programme suivant. On obtient le résultat de la FIGURE 1.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  #tableau des valeurs mesurees
5  T5_data = np.array([3.78, 3.84, 3.91, 3.84, 3.91, 3.81, 3.81, 3.85, 3.82, 3.82])
6
7  #tracé de l'histogramme
8  plt.figure()
9  plt.hist(T5_data)
10 plt.xlabel('valeurs mesurées') #légende de l'axe des abscisses
11 plt.ylabel('occurrences') #légende de l'axe des ordonnées
12 plt.show()
```



Dans la FIGURE 1 on a tracé un histogramme des données de **l'exemple 1** :

- sur l'axe des abscisses l'intervalle des valeurs mesurées est divisé en un nombre d'intervalles de même largeur appelés **classes** (ici il y a 10 classes, ce nombre étant calculé automatiquement par la fonction `np.hist`) ;
- le **nombre de valeurs mesurées dans chaque classe** est représenté par un **rectangle.**

FIGURE 1. Histogramme des valeurs mesurées pour T_5.

IV Simulation Monte-Carlo

☞ On appelle simulation Monte-Carlo une procédure basée sur la **génération de nombres aléatoires** permettant de **simuler la variabilité d'une mesure**. On l'utilisera en TP soit parce que la variabilité est cachée (si la précision de l'instrument de mesure est trop faible par exemple), soit parce qu'on n'a pas le temps de répéter plusieurs fois la mesure.

1) Exemple d'une seule mesure

Exemple 3 (Mesure d'une masse avec une balance)

On a mesuré la masse m d'un cylindre en métal à l'aide d'une balance à affichage numérique, dont la précision est 1 g. La balance affiche 100 g.

✍ À l'aide d'une simulation Monte-Carlo, calculer l'incertitude-type associée à cette mesure.

→ **Réponse.** L'idée est de **simuler** la variabilité cachée par la résolution de la balance. La précision de la balance étant de 1 g, il est raisonnable de considérer que la valeur de la masse à mesurer se trouve sûrement dans l'intervalle [99,5 g ; 100,5 g] (soit une demi-étendue $\Delta = 0{,}5$ g). Le code à la page suivante simule $k = 100\,000$ mesures donnant un résultat dans cette intervalle, avec une **distribution uniforme** (aucune valeur n'est plus probable qu'une autre dans cette intervalle). **L'incertitude-type de la mesure avec la balance** s'obtient en calculant **l'écart-type** pour la série de valeurs simulées.

Dans la ligne `12` on a utilisé la fonction `np.random.uniform(a,b,k)` du module `numpy`, qui génère k nombres aléatoires dans l'intervalle $[a, b]$ selon une distribution uniforme.

La variable `m_sim` dans la ligne `12` stocke donc un tableau de $k = 100\,000$ valeurs tirées aléatoirement dans l'intervalle $[99{,}5\,\text{g}; 100{,}5\,\text{g}]$. On peut calculer son écart-type, comme on l'a fait à la ligne `15`. On peut aussi tracer son histogramme (lignes ligne `18` à ligne `22`). On obtient le résultat de la FIGURE 2.

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

#masse mesurée
m = 100  #g
# demi-étendue de la plage de mesures
Delta = 0.5 #g
#nombre de mesures simulées
k = 100000

#tirage aléatoire de k valeurs avec une distribution uniforme dans [m-Delta,m+Delta]
m_sim = np.random.uniform(m-Delta,m+Delta,k)

#calcul de l'écart-type
u_m = np.std(m_sim)

#tracé de l'histogramme
plt.figure()
plt.hist(m_sim, bins = 20)
plt.xlabel('valeurs simulées') #légende de l'axe des abscisses
plt.ylabel('occurrences') #légende de l'axe des ordonnées
plt.show()
```

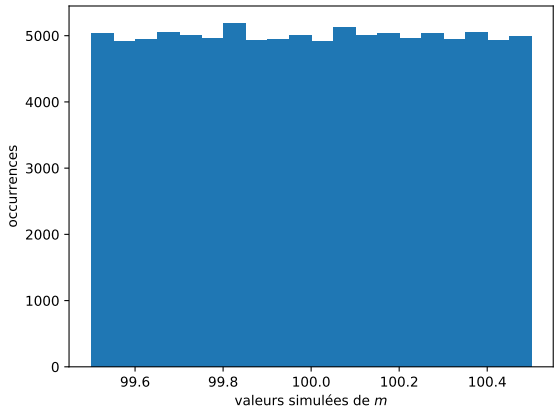


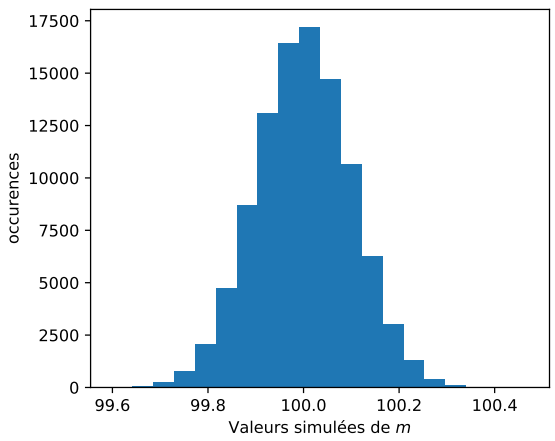
FIGURE 2. Histogramme des valeurs simulées pour m.

Dans la FIGURE 2 on a tracé un histogramme des valeurs simulées pour la mesure de **l'exemple 3** :

- sur l'axe des abscisses l'intervalle des valeurs simulées est divisé en 20 classes (nombre choisi en passant le paramètre optionnel `bins = 20` dans `plt.hist`) ;
- toutes les classes contiennent un nombre similaire de valeurs, aux fluctuations aléatoire près ce qui est caractéristique d'une distribution uniforme.

Noter que le résultat du calcul de l'écart-type donne pour `u_m` la valeur 0,2890 g, à comparer à la formule théorique $u(m) = \Delta/\sqrt{3}$ qui donne $u(m) = 0{,}2886$ g. L'accord est excellent.

Remarque importante : on peut **utiliser d'autres distributions de nombres aléatoires** selon le contexte de la mesure. Si on pense que la distribution des valeurs à simuler devrait être piquée et pas uniforme on peut utiliser la fonction `np.random.normal(mu, sigma, k)` qui renvoie `k` valeurs selon une **distribution gaussienne** (courbe en cloche) centrée en `mu` (valeur moyenne) et de largeur `sigma` (écart-type). Cette possibilité est détaillée ci-dessous.




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

#masse mesurée
m = 100  #g
# largeur de la distribution gaussienne
sigma = 0.1 #g (jugement expérimental)
#nombre de mesures simulées
k = 100000
#tirage aléatoire de k valeurs (gaussienne)
m_sim = np.random.norma(m,sigma,k)
#tracé de l'histogramme
plt.figure()
plt.hist(m_sim, bins = 20)
plt.xlabel('valeurs simulées')
plt.ylabel('occurrences')
plt.show()
```

2) Exemple pour la propagation des incertitudes

Exemple 4 (Mesure de l'aire d'une table)

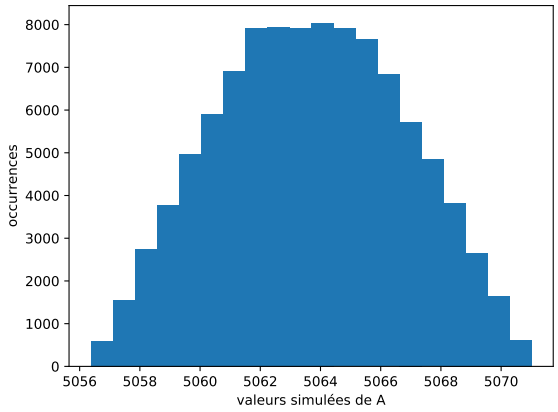
On mesure les dimensions d'une table (longueur L et largeur ℓ) à l'aide d'un mètre ruban enrouleur gradué au mm. On mesure $L = 91{,}9$ cm et $\ell = 55{,}1$ cm.

✍ À l'aide d'une simulation Monte-Carlo estimer l'incertitude-type $u(A)$ sur l'aire $A = L \times \ell$.

→ **Réponse.** L'idée est de **simuler** la variabilité cachée par la résolution du mètre ruban. La précision du mètre ruban étant de 1 mm, il est raisonnable de considérer que la demi-étendue des plages de mesure des longueurs est $\Delta = 0{,}5$ mm $= 0{,}05$ cm. Le code suivant simule $k = 100\,000$ mesures de L et ℓ et calcule l'aire A pour chaque couple. On obtient une distribution de k valeurs simulées pour A. **L'incertitude-type de la mesure** de A s'obtient en calculant **l'écart-type** de la série de valeurs simulées.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  #longueurs mesurées
5  L = 91.9  #cm
6  l = 55.1 #cm
7  # demi-étendue de la plage de mesures
8  Delta = 0.05 #cm
9  #nombre de mesures simulées
10 k = 100000
11 #tirage aléatoire de k valeurs avec une distribution uniforme
12 L_sim = np.random.uniform(L-Delta,L+Delta,k)
13 l_sim = np.random.uniform(l-Delta,l+Delta,k)
14
15 #calcul de l'aire A = L x l pour chaque couple de valeurs simulées
16 A_sim = L_sim*l_sim
17
18 #calcul de l'écart-type
19 u_A = np.std(A_sim)
20 #tracé de l'histogramme
21 plt.figure()
22 plt.hist(A_sim, bins = 20)
23 plt.xlabel('valeurs simulées') #légende de l'axe des abscisses
24 plt.ylabel('occurrences') #légende de l'axe des ordonnées
25 plt.show()
```

Les lignes 12 et 13 génèrent deux tableaux contenant 100 000 valeurs aléatoires de L et ℓ et la ligne 16 forme 100 000 produits $A = L \times \ell$ à partir de ces valeurs. Le tableau `A_sim` contient donc 100 000 valeurs simulées de l'aire de la table, dont l'histogramme est représenté dans la FIGURE 3.



Dans la FIGURE 3 on a tracé un histogramme des valeurs simulées pour la mesure de **l'exemple 4** :

- sur l'axe des abscisses l'intervalle des valeurs simulées est divisé en 20 classes (nombre choisi en passant le paramètre optionnel `bins = 20` dans `plt.hist`) ;
- la distribution obtenue a une forme de pyramide aplatie indiquant que les valeurs extrémales sont moins probables que les valeurs centrales.

FIGURE 3. Histogramme des valeurs simulées pour A.

La valeur de `u_A` dans la ligne 22 est 3,0989 cm^2, ce qui permet de vérifier la validité de la formule théorique

$$u(A) = A \times \sqrt{\left(\frac{u(L)}{L}\right)^2 + \left(\frac{u(\ell)}{\ell}\right)^2} \quad \text{donnant} \quad 3{,}0927\,\text{cm}^2., \tag{1}$$