[Méthode] Simulations Monte Carlo sous Python

Il est souvent impossible expérimentalement d'observer la variabilité d'une mesure sur un très grand nombre d'obervations. Connaissant l'incertitude de mesure, on peut en simuler *in silico* un nombre aussi important que l'on souhaite, en utilisant un algorithme de génération de nombres alétoires. C'est le principe des méthodes Monte Carlo, en référence aux casinos où le hasard joue un grand rôle.

Génération des échantillons

Les valeurs aléatoires seront stockées sous Python sous forme de tableaux *numpy*. On importe donc les bibliothèques suivantes :

```
import numpy as np
import numpy.random as rd
```

Ensuite, on génère des échantillons sous forme de tableaux (notés ici x_{MC}) en précisant la loi de distribution désirée :

— pour générer N valeurs selon une loi de probabilité Gaussienne de moyenne $\boldsymbol{m}$ et d'écart-type $\boldsymbol{u}$, on utilise :

```
x_MC = m + u * rd.normal(0, 1, N)
```

— pour générer N valeurs selon une loi de probabilité uniforme de moyenne $\boldsymbol{m}$ et de demi-largeur $\boldsymbol{d}$, on utilise :

```
x_MC = m + d * rd.uniform(-1, 1, N)
```

Utilisation pour les incertitudes composées

Lorsque la grandeur d'intérêt est calculée à partir d'autres grandeurs, on peut utiliser la simulation Monte Carlo en générant des échantillons de même taille de chacune des grandeurs mesurées. Tous les éléments au même rang dans le tableau constituent alors une expérience simulée.

Il suffit alors d'effectuer le calcul directement sur les tableaux *numpy* générés, le calcul se fait rang par rang et produit un nouveau tableau *numpy* de même taille que les données, qui contient les calculs pour toutes les expériences simulées.

Par exemple pour calculer une aire, les tableaux L_{MC} et l_{MC} correspondant à des échantillons simulées, on utilise directement :

```
A_MC = L_MC * l_MC
```

Visualisation de l'histogramme de variabilité

On importe la bibliothèque graphique :

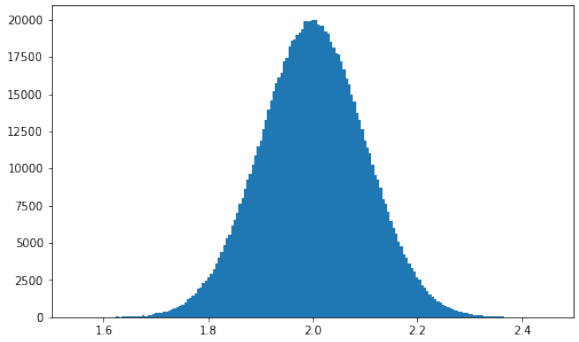
```
import matplotlib.pyplot as plt
```

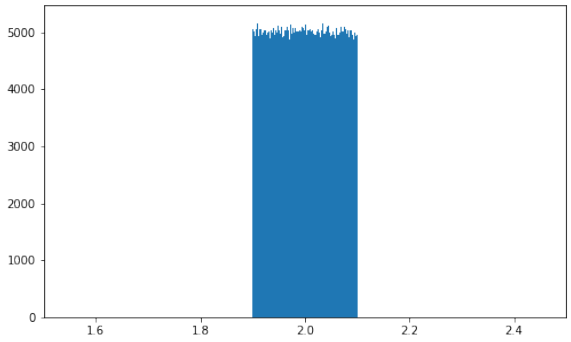
Pour afficher un histogramme du tableau x_{MC}, le code est :

```
plt.hist(x_MC, bins='rice')
plt.show()
```

Note : l'option *bins='rice'* permet d'optimiser la largeur des intervalles utilisés.

Pour les deux distributions précédentes, avec $N = 1\,000\,000$, on obtient les résultats suivants :





Affichage des caractéristiques de la distribution

La moyenne et l'écart-type d'un échantillon se calculent à l'aide des commandes suivantes :

```
moy_x = np.mean(x_MC)
u_x = np.std(x_MC, ddof=1)
print("Valeur mesurée : x =", moy_x)
print("Incertitude-type : u(x) =", u_x)
```