# ON 2 - Evaluation d'une incertitude-type par la méthode de Monte Carlo

La variabilité d'une mesure expérimentale est liée à de nombreux facteurs tel que l'instrument de mesure utilisé, le choix de la méthode de mesure, les paramètres extérieurs. On quantifie la dispersion des résultats de mesure par une grandeur mathématique, appelée **écart-type.**

**L'incertitude-type est par définition l'écart type de la distribution des résultats obtenus lors d'un grand nombre de mesures.**

- On parle d'**évaluation de type A lorsque** la mesure est répétée plusieurs fois, l'expérimentateur obtient plusieurs valeurs de la grandeur mesurée. On fait la moyenne des valeurs obtenues et on calcule l'écart-type de cet ensemble de valeurs.

- Dans le cas d'une **évaluation de type B,** l'expérience n'est pas répétée plusieurs fois, il faut mettre en œuvre une autre méthode pour évaluer l'incertitude-type (*voir Fiche méthode Incertitudes*).

  On peut notamment utiliser la **méthode de Monte-Carlo qui consiste à simuler numériquement la répétition de l'expérience**. **L'écart-type de l'ensemble des valeurs obtenues lors de ces simulations fournit l'incertitude-type recherchée**.

  Cette méthode est très utile dans le cas d'une propagation d'incertitudes complexe**,** c'est-à-dire lorsque l'on souhaite déterminer l'incertitude-type sur une grandeur calculée à partir de grandeurs expérimentales, mesurées ou fournies.

## I. Méthode Monte-Carlo

### 1. Exemple : détermination de la concentration d'une solution fille lors d'une dilution

On réalise la dilution d'une solution mère de concentration $C_m$ = 1 mol/L (on négligera l'incertitude sur cette valeur). On utilise le matériel suivant :


On note $C_f$ la concentration de la solution obtenue, $V_f$ son volume et $V_m$ le volume de solution mère prélevé.

La concentration de solution fille ainsi obtenue est : $C_f = \frac{C_m V_m}{V_f}$

On choisit d'utiliser une simulation Monte Carlo pour déterminer l'incertitude.

## 2. Principe de la méthode

1. **Lister les grandeurs expérimentales** utiles pour le calcul et associer à chacune, un intervalle au sein duquel on peut raisonnablement penser que celle-ci appartient.
2. Procéder à un **tirage au sort aléatoire** d'un jeu de valeurs pour chaque grandeur expérimentale dans l'intervalle estimé.
3. **Calculer** la grandeur recherchée avec ce jeu de valeurs.
4. **Stocker** les valeurs dans une liste de résultats.
5. **Calculer l'écart-type** de l'ensemble des valeurs obtenues : **le résultat est l'incertitude-type**.

## 3. Fonctions pour réaliser le tirage au sort

La bibliothèque `numpy` est ici utilisée pour simuler un processus aléatoire (`np.random`). On peut envisager 3 cas de figure pour réaliser le tirage au sort de N valeurs contenue dans un intervalle :

→ **Distribution uniforme : probabilité identique sur tout l'intervalle**, on utilise la commande :

```
np.random.uniform(borne_inf,borne_sup, nombre de valeurs)
```

→ **Distribution triangulaire : probabilité plus forte pour une valeur au centre d'un intervalle**, on utilise la commande :

```
np.random.triangular(borne_inf,centre,borne_sup, nombre de valeurs).
```

→ **Distribution normale** : **probabilité suivant une loi normale**, si on connaît l'écart-type, on utilise la commande :

```
np.random.normal(valeur moyenne,écart-type, nombre de valeurs).
```

## 4. Exemple de code

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

#Entrée des données
Cm=1
Vm=25
Vf=100
delta_Vm=0.03
delta_Vf=0.10

#Simulation répétée N fois
N=10000         #Nombre de simulations
Cf_sim = []       #Liste vide pour stocker les valeurs calculées Cf
Vm_sim = np.random.uniform(Vm-delta_Vm, Vm+delta_Vm,N) #Tire N valeurs de
Vm selon une loi uniforme centrée sur Vm et de précision delta_Vm
Vf_sim = np.random.uniform(Vf-delta_Vf, Vf+delta_Vf,N)
Cf = Cm*Vm_sim/Vf_sim
Cf_sim.append(Cf)

#Calcul de l'écart-type expérimental
uCf = np.std(Cf_sim,ddof=1)
```

```
#Affichage des résultats avec 5 chiffres après la virgule
print("u(Cf) = {:.5f} mol/L".format(uCf))

#Génération d'un histogramme (optionnel)
plt.hist(Cf_sim,bins='rice') # La commande 'rice' permet d'optimiser les
intervalles d'affichage de l'histogramme
plt.show()

u(Cf) = 0.00023 mol/L
```