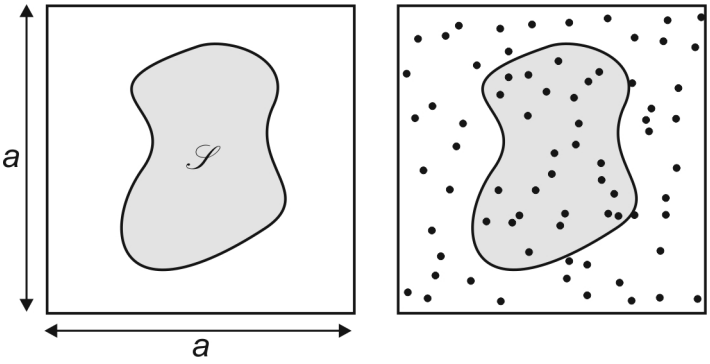
Capacité numérique : simuler, à l'aide d'un langage de programmation ou d'un tableur, un processus aléatoire permettant de caractériser la variabilité de la valeur d'une grandeur composée.

1. Introduction : simulation de Monte Carlo

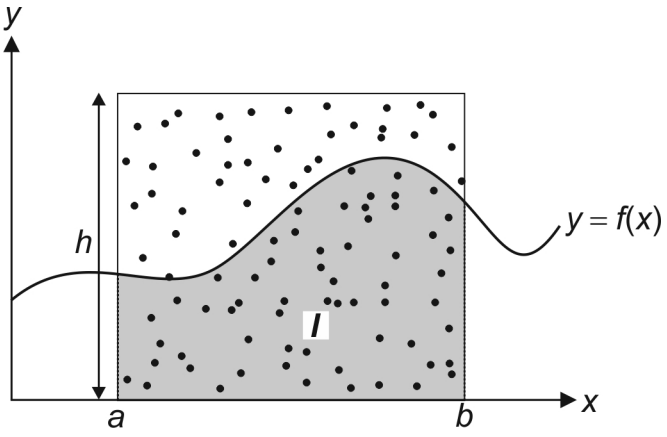
On cherche à mesurer l'aire $\mathscr{S}$ d'un lac. On tire pour cela de façon aléatoire N boulets de canon sur une surface carrée a^2 contenant le lac On suppose que la distribution de probabilité est *uniforme* sur le carré, ce qui veut dire que la probabilité que le boulet tombe en un point du carré est la même en tout point.



Si n boulets tombent dans le lac, le rapport n/N donne une approximation du rapport $\mathscr{S}/a^2$, d'autant meilleure que N est grand (loi des grands nombres), et on en déduit $\mathscr{S} = \dfrac{n}{N}a^2$.

Cette méthode, appelée *méthode de Monte-Carlo*, permet donc de calculer des intégrales (comme celle qui définit l'aire $\mathscr{S}$ précédente).

Pour une fonction réelle f d'une seule variable réelle x, on peut calculer $I = \int_a^{b>a} f(x)\mathrm{d}x$ en effectuant N tirages aléatoires de points $M(x,y)$ avec $x \in [a,b]$ et $y \in [0,h]$, où h est plus grand que le maximum des valeurs $f(x)$ pour $x \in [a,b]$. La distribution de probabilité est uniforme sur le rectangle $(x,y) \in [a,b]\times[0,h]$.



Là encore, si n est le nombre de points vérifiant $y < f(x)$, on a $I \simeq \dfrac{n}{N}h(b-a)$. On se rapproche de la valeur réelle en prenant N grand (mais l'écart diminue lentement avec N tout en fluctuant aléatoirement…).

2 Génération de valeurs selon une densité de probabilité donnée

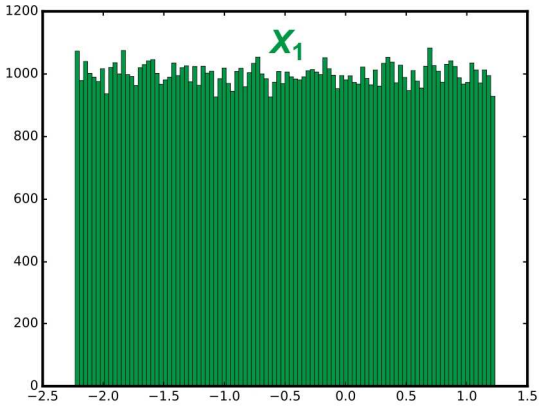
Sous Python, on utilise ici la librairie **`random`** de **`numpy`** et on trace des histogrammes avec **`plt.hist`**.

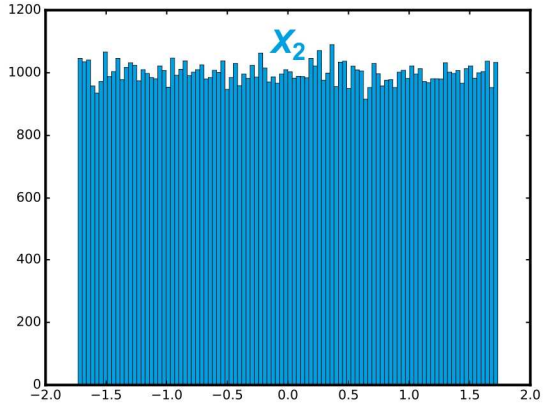
Dans l'exemple ci-après, on a généré 1000 tirages aléatoires de grandeurs X_1, X_2 et X_3 de densité de probabilité rectangulaire d'écart-type égal à 1 (donc de demi-largeur égale à $\sqrt{3}$), respectivement centrées sur $-0{,}5$, 0 et 0,5. Par exemple pour X_1 on génère 1000 valeurs comprises entre $-0{,}5-\sqrt{3} \simeq -2{,}23$ et $-0{,}5+\sqrt{3} \simeq +1{,}23$.

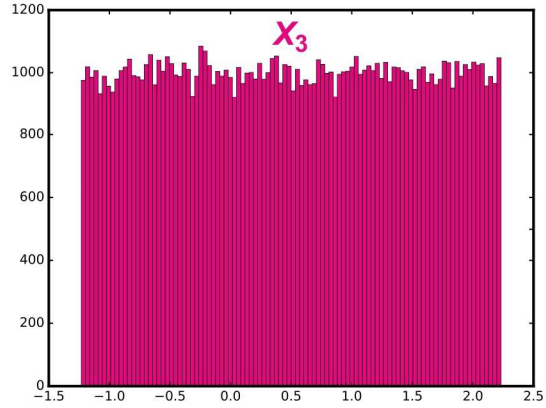
On obtient pour X_1 les tirages sous forme d'histogramme ainsi :

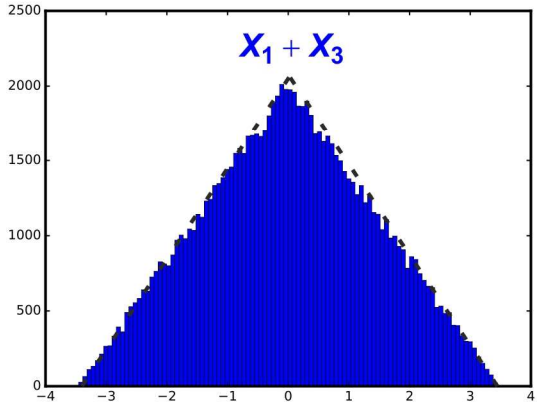
```
import numpy as np #
import numpy.random as rd # on va ici utiliser la bibliothèque random de numpy
from matplotlib import pyplot as plt

moy1 = -0.5
sigma1 = 1
xmin1,xmax1 = moy1-sigma1*3**.5,moy1+sigma1*3**.5 # on calcule les bornes de l'inter-
valle (facteur racine de 3 entre le demi-intervalle et l'écart-type)
N = 100000
X1 = rd.uniform(xmin1,xmax1,N) # tire aléatoirement N nombres autour de moy avec un
écart-type sigma (renvoie un tableau 1D)
plt.hist(X1,100,color='g') # le nombre d'intervalles de l'histogramme est fixé à 100
plt.show()
>>> np.mean(X1) # moyenne du tableau de valeurs
-0.00079578738478856879 #on vérifie que la valeur moyenne est bien proche de 0
>>> np.std(X1,ddof = 1) # écart-type du tableau de valeurs
1.0004637660849285 #on vérifie que l'écart-type est bien proche de 1.
```

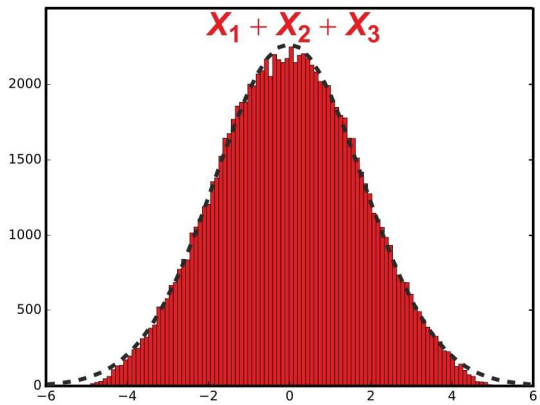
On recommence pour les variables X_2, X_3, $X_1 + X_3$ et $X_1 + X_2 + X_3$. On constate que $X_1 + X_3$ est distribuée selon une loi triangulaire et $X_1 + X_2 + X_3$ quasiment selon une loi normale (dans ce cas particulier, le théorème central limite donne une bonne approximation pour la somme de trois variables, même si leur valeur moyenne est différente).











3 Propagation des incertitudes

Pour déterminer sous Python l'incertitude-type sur $X = f(X_1, X_2, ..., X_k, ...)$, connaissant les types de distribution, moyennes et écarts-type sur $X_1, X_2, ..., X_k, ...$:

— Si la distribution de X_k est normale, elle est définie par la liste :

`Xk = ['normal',mu,sigma]`, où `mu` et `sigma` sont la moyenne et l'écart-type de la distribution.

— Si la distribution de X_k est rectangulaire, elle est définie par la liste :

`Xk = ['rect',mu,sigma]`, où `mu` et `sigma` sont toujours la moyenne et l'écart-type de la distribution.

— On génère $N = 1\,000\,000$ tirages aléatoires de X_k obéissant à la distribution choisie, à l'aide de `np.random.normal(Xk[1],Xk[2],N)` et de `np.random.uniform(xmin,xmax,N)` selon les cas. `xmin,xmax`, valeurs extrêmes de la distribution rectangulaire, sont calculés par :

`xmin,xmax = Xk[1]-Xk[2]*3**.5,Xk[1]+Xk[2]*3**.5`.

> La grande valeur N de tirages est préconisée pour rendre négligeables les fluctuations entre deux tirages différents mais peut être réduite par exemple à 10 000 si les calculs sont trop longs.

`N` est passée en variable globale.

— Les distributions obtenues (tableaux 1D `numpy` de type `array`) sont stockées dans une liste `L`.

— On applique la fonction `f` passée en variable globale aux éléments de `L`. Elle renvoie la distribution `X`.

— On calcule et on renvoie la valeur moyenne et l'écart-type de `X`. On affiche également son histogramme.

```
N = 1000000

def propagation_monte_carlo(*arg):
    L = []
    for elt in arg:
        if elt[0] == 'normal':
            L.append(rd.normal(elt[1],elt[2],N))
        else:
            xmin,xmax = elt[1]-elt[2]*3**.5,elt[1]+elt[2]*3**.5
            L.append(rd.uniform(xmin,xmax,N))
    X = f(*L)
    mu = np.mean(X)
    sigma = np.std(X, 0, ddof = 1)
    plt.hist(X,100)
    plt.show()

    return mu,sigma
```

Dans le cas de relations linéaires, par exemple $X = X_1 + X_2$, la méthode analytique fournit le bon résultat quel que soit le type de distributions.

Exemple : on étudie à l'oscilloscope la réponse indicielle d'un passe-bas du second ordre. On mesure le facteur de qualité $Q = 4{,}99$ et la pseudo-période des oscillations $T = 990\ \mu s$. Les incertitudes-type mesurées sont $u(Q) = 0{,}84$ et $u(T) = 120\ \mu s$. Les distributions sont uniformes.

On cherche à déterminer la fréquence des oscillations libres non amorties $f_0 = \dfrac{1}{T\sqrt{1-\dfrac{1}{4Q^2}}}$

et son incertitude-type.

(1) Appliquons la méthode analytique : $f_0 = \dfrac{1}{T\sqrt{1-\dfrac{1}{4Q^2}}} = 1015{,}2\ \text{Hz}$

Pour l'incertitude-type, on pose $q = 1-\dfrac{1}{4Q^2} = 0{,}9900$

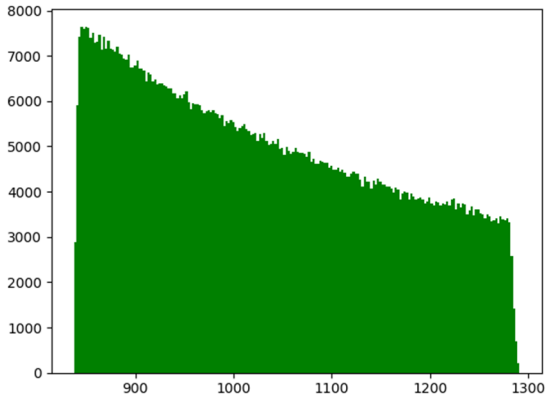
$$\Rightarrow \mathrm{d}q = \frac{1}{2Q^3}\mathrm{d}Q \Rightarrow u(q) = \frac{u(Q)}{2Q^3} = 3{,}38\cdot 10^{-3}.$$

On a alors $f_0 = \dfrac{1}{T\sqrt{q}} \Rightarrow \ln(f_0) = -\ln(T) - \dfrac{1}{2}\ln(q) \Rightarrow \dfrac{\mathrm{d}f_0}{f_0} = -\dfrac{\mathrm{d}T}{T} - \dfrac{1}{2}\dfrac{\mathrm{d}q}{q}$

$$\Rightarrow \frac{u(f_0)}{f_0} = \sqrt{\left[\frac{u(T)}{T}\right]^2 + \frac{1}{4}\left[\frac{u(q)}{q}\right]^2} = 0{,}121 \ :\ u(f_0) = 123\ \text{Hz}$$

(2) Réalisons une simulation de Monte-Carlo :

```
def f(T,Q):
    return 1/(T*(1-1/4/Q**2)**0.5)

T = ['rect',990E-6,120E-6]

Q = ['rect',4.99,0.84]

>>> propagation_monte_carlo(T,Q)
(1031.0292838300993, 127.24894324531354)
```



On a donc $f_0 = 1030\ \text{Hz}$; $u(f_0) = 130\ \text{Hz}$

L'effet de la non-linéarité est maintenant visible, et d'autant plus que les écarts-type sur les X_k sont importants. La méthode analytique peut donner un résultat et un écart-type erronés (en rouge). D'autre part, la distribution des valeurs de f_0 s'écarte notablement de la loi normale.

On remarque également que borner le résultat de la mesure (ici $f_0 \in [837\ \text{Hz}, 1292\ \text{Hz}]$) est sans intérêt puisque la distribution n'est pas rectangulaire : on n'en tire aucune information sur la valeur moyenne ni sur l'écart-type.

Pour des relations non linéaires entre X et les X_k et des écarts-type importants, la méthode de Monte-Carlo donne *sans calcul* des résultats *justes* pour $u(X)$ contrairement à la méthode analytique.