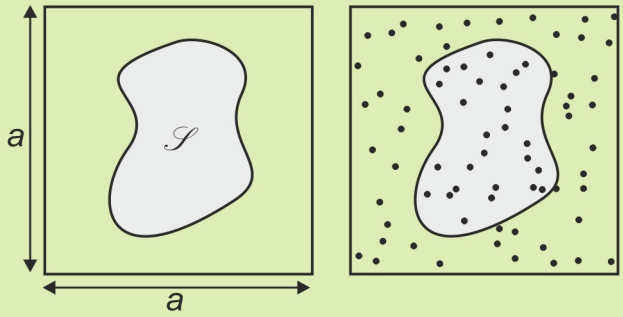
Capacité numérique : simuler, à l'aide d'un langage de programmation ou d'un tableur, un processus aléatoire permettant de caractériser la variabilité de la valeur d'une grandeur composée.

Méthode numérique : simulation de Monte-Carlo

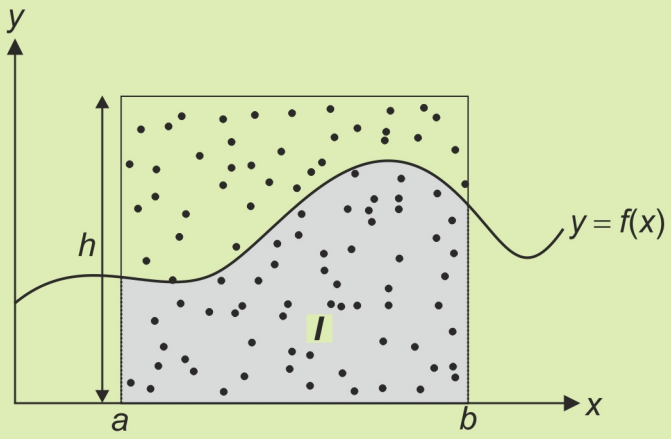
On cherche à mesurer l'aire $\mathscr{S}$ d'un lac. On tire pour cela de façon aléatoire N boulets de canon sur une surface carrée a^2 contenant le lac. On suppose que la distribution de probabilité est *uniforme* sur le carré, ce qui veut dire que la probabilité que le boulet tombe en un point du carré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

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 $I \simeq \frac{n}{N}h(b-a)$. On se rapproche de la valeur réelle en prenant N grand (mais l'écart diminue lentement avec N, tout en fluctuant aléatoirement…).

Générer des valeurs selon une densité de probabilités donnée

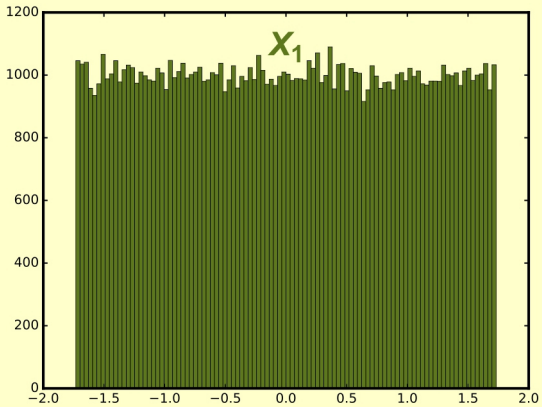
Sous Python, on utilise ici la librairie **`random`** de **`numpy`** et on trace des histogrammes avec **`plt.hist`**.

Dans l'exemple ci-après, on a généré 100 000 tirages aléatoires de la grandeur X_1 de densité de probabilité rectangulaire d'écart-type égal à 1 (donc de demi-largeur égale à $\sqrt{3}$), centrée sur 0, soit 100 000 valeurs comprises entre $-\sqrt{3}$ et $\sqrt{3}$.

On obtient 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
sigma1 = 1
xmin1,xmax1 = moy1-sigma1*3**.5,moy1+sigma1*3**.5 # on calcule les
bornes de l'intervalle (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 proche de 1
```

Pour une distribution normale (gaussienne) d'écart-type égal à 1, on utilise :

```
X1 = np.random.normal(moy1,sigma1,N) # tire aléatoirement N nombres
autour de moy1 avec un écart-type sigma1 (renvoie un tableau 1D)
```

Q.1) Effectuer de même 100 000 tirages pour les grandeurs X_2 et X_3, également de densités de probabilité rectangulaire d'écart-type égal à 1, respectivement centrées sur $-0{,}5$ et 0,5.

Représenter les histogrammes des variables X_2, X_3, $X_2 + X_3$ et $X_1 + X_2 + X_3$. Ces variables obéissent-elles également à une distribution rectangulaire ? Proposer une explication.

Propagation des incertitudes : méthode analytique

La mesure d'une grandeur X fournit un résultat $\overline{x}$ avec une incertitude-type $u(X)$. Si la grandeur X n'est pas directement accessible par la mesure mais liée à des mesures *indépendantes* de *deux grandeurs elles-mêmes indépendantes* X_1 et X_2 dont les valeurs mesurées sont $\overline{x_1}$, avec une incertitude-type $u(X_1)$ et $\overline{x_2}$, avec une incertitude-type $u(X_2)$. La relation s'écrit sous la forme $X = f(X_1, X_2)$.

Si les incertitudes sont suffisamment petites, on montre que l'on a $\overline{x} = f(\overline{x_1}, \overline{x_2})$ et :

$u(X) = \sqrt{u^2(X_1)\left[\frac{\partial f}{\partial X_1}(\overline{x_1}, \overline{x_2})\right]^2 + u^2(X_2)\left[\frac{\partial f}{\partial X_2}(\overline{x_1}, \overline{x_2})\right]^2}$. Cette relation peut se généraliser au cas où X dépend de plus de deux variables : $\overline{x} = f(\overline{x_1}, \overline{x_2}, ..., \overline{x_k}, ...)$ et $u(X) = \sqrt{\sum_k \left[\frac{\partial f}{\partial X_k}\right]^2 u^2(X_k)}$.

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.

Les fonctions de distributions des grandeurs X_k peuvent être différentes, et certaines incertitudes-type évaluées par des méthodes de type A, d'autres par des méthodes de type B.

Propagation des incertitudes : méthode de Monte-Carlo

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
```

Q.2) 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 sont $u(Q) = 0{,}84$ et $u(T) = 120\ \mu s$. Les distributions sont uniformes.

Déterminer la fréquence des oscillations libres non amorties $f_0 = \dfrac{1}{T\sqrt{1-\dfrac{1}{4Q^2}}}$ et son incertitude-type :

— Par la méthode analytique

— En utilisant la méthode de Monte-Carlo.

Comparer les résultats fournis par les deux méthodes.

Tracer l'histogramme des valeurs de f_0. Cette distribution est-elle uniforme ? Peut-on tirer des renseignements sur les valeurs extrêmes prises par f_0 ?

Reprendre la comparaison si $u(Q) = 0{,}12$ et $u(T) = 40\ \mu s$. Commenter.