

TP modélisation MESP 803 : Traitement des données

Objectifs :

1. générer des nombres aléatoires suivant des distributions uniformes ou normales.
2. renforcer la panoplie des procédures d'affichage : histogrammes, plusieurs graphes sur une même figure, courbes avec barres d'erreurs...
3. écrire et lire des données numériques dans un fichier ASCII suivant un format donné.
4. maîtriser la méthode des moindres carrés pour les régressions linéaires.
5. utilisation de la bibliothèque d'ajustement de courbes (fits) sur des relations linéaires et non linéaires.

A Nombres aléatoires

Le principe de fonctionnement des calculateurs modernes repose sur des algorithmes déterministes. Il est alors *a priori* illusoire de penser pouvoir générer une suite de nombres aléatoires comme on pourrait l'imaginer à partir d'une expérience comme un lancé de dés. Néanmoins, les mathématiciens ont mis au point des algorithmes qui présentent tous les aspects d'un comportement régi par le pur hasard... au moins sur des séries finies de nombres ; la récurrence de ces séries n'apparaissant, en pratique, qu'au bout d'un nombre très grand de tirages. Nous n'allons pas étudier ce type d'algorithme d'un point de vue mathématique mais les utiliser pour générer des séries de nombres aléatoires (pseudo-aléatoires devrait-on dire) qui permettent alors de simuler le bruit inhérent à toute mesure d'une grandeur physique.

A.1 Distribution uniforme

Exercice n° 1 Sur l'intervalle $[0, 1]$

C'est la loi de distribution de base à partir de laquelle on peut engendrer toutes les autres. La bibliothèque `scipy` contient la fonction `rand(Ne)` qui renvoie un tableau de N_e éléments répartis uniformément dans l'intervalle $[0, 1]$.

1. Générer une séquence de tels nombres et les représenter dans un premier temps à l'aide de la fonction `plot` de `pylab`.
2. Il est souvent utile de représenter une suite statistique de données sous forme d'histogramme. La fonction `hist()` de `pylab` permet de réaliser cette opération ; en argument on passe obligatoirement le tableau des données (ici le tableau généré

précédemment par la fonction `rand`) et de façon optionnel : le nombre de « boîtes » de regroupement que l'on désire, ainsi que l'option `normed=True` si on veut normaliser la distribution (en général pour la comparer à une distribution théorique). Rechercher dans la documentation de `matplotlib` la syntaxe exacte de `hist()` puis effectuer cette représentation.

3. En utilisant la fonction `subplot()` de `pylab` disposer ces deux représentations sur la même figure.

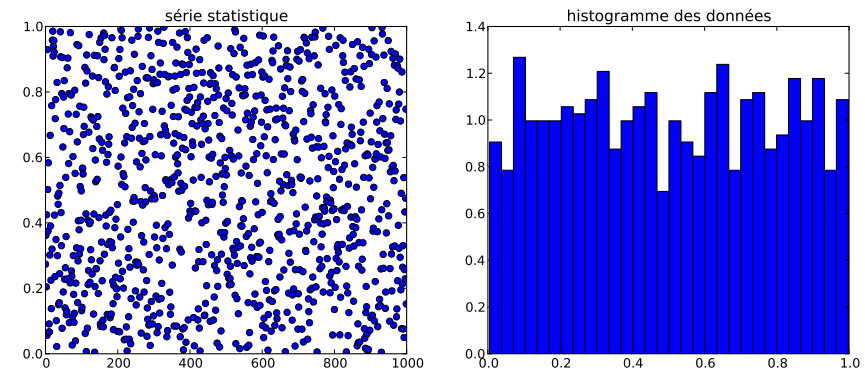


FIGURE 1 – Représentation d'une série statistique de données réparties uniformément sur l'intervalle $[0, 1]$

Exercice n° 2 Sur un intervalle $[a, b]$

En dilatant l'amplitude de l'intervalle d'un facteur $(b - a)$ et translatant l'origine de 0 à a par l'opération $(b-a)*\text{rand}(N_e)+a$, on engendre ainsi N_e échantillons distribués uniformément dans l'intervalle considéré.

1. Représenter sous forme d'histogramme un échantillonnage X de données réparties uniformément dans l'intervalle $[-\alpha, \alpha]$.
2. Calculer la valeur moyenne $\langle X \rangle$ sur cet échantillonnage ainsi que l'écart type $\sigma = \sqrt{\langle (X - \langle X \rangle)^2 \rangle}$. Vérifier que lorsque N_e devient « assez grand » $\langle X \rangle$ tend vers zéro et l'écart type σ tend vers la valeur $\frac{\alpha}{\sqrt{3}}$ (sauriez-vous le démontrer?).

A.2 Distribution normale

On montre en mathématique que pour une « bonne » distribution statistique d'une variable aléatoire X et d'écart type σ , on peut construire une nouvelle variable aléatoire

$$Y = \frac{\sum_{i=1}^N X_i}{\sqrt{N}} \quad (1)$$

telle que lorsque N tend vers l'infini Y obéit à la loi de distribution normale (ou gaussienne)

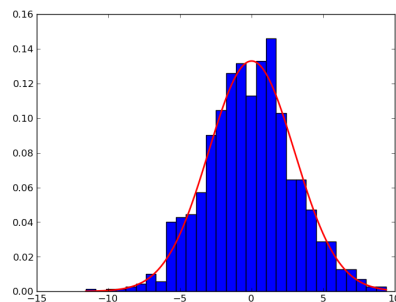
$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left[-\frac{(y - Y_m)^2}{2\sigma^2} \right] \quad (2)$$

où $Y_m = \langle Y \rangle$ est la valeur moyenne de Y et vaut $\sqrt{N} \langle X \rangle$.

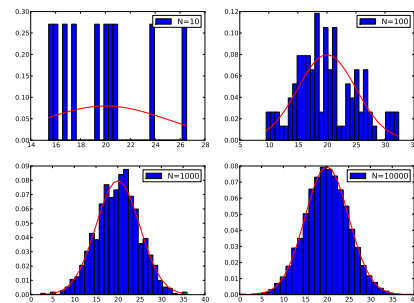
En pratique nous allons voir que pour N de l'ordre de seulement quelques unités, la distribution statistique de Y s'apparente très rapidement à la distribution gaussienne.

Exercice n° 3 Génération « maison » d'une loi normale

1. Créer tout d'abord une fonction `rand_maison(Np, sigma)` générant un tableau de N nombres aléatoires distribués uniformément sur un intervalle de valeur moyenne nulle et d'amplitude telle que son écart type corresponde à une valeur σ donnée.
2. Construire alors à partir de cette fonction un tableau de valeur Y engendré suivant la relation (1) et représenter sa distribution sous forme d'histogramme.
3. Comparer avec la relation théorique (2) en traçant sa représentation sur le même graphe.
4. Calculer l'écart type pour votre échantillonnage et comparer avec la valeur attendue (faites varier le nombre de points regroupés, N , ainsi que le nombre d'échantillons N_e).



(a) Fabrication « maison »



(b) Loi normale de numpy

FIGURE 2 – Distributions normales

Exercice n° 4 Génération avec numpy d'une loi normale

1. On dispose directement dans la bibliothèque `numpy` de la fonction `normal(Xm, sigma, Ne)` qui retourne un tableau de N_e nombre aléatoires centrés sur la valeur

moyenne X_m et d'écart type σ . Utilisez-la et comparez avec votre programme « maison ».

2. Tracer différentes distributions en prenant $N_e = 10, 100, 1000, 10000$ et les placer sur une même figure en utilisant la fonction `subplot()`.

Exercice n° 5 Création d'un fichier de données simulant une expérience

On remplit un volume V constant d'un gaz supposé parfait sous une pression de 1 bar et à la température de 20°C

1. Question préliminaire : Exprimer la relation entre la température θ (en °C) et la pression (en millimètres de mercure (mm Hg)¹) sous la forme $\theta = AP + B$. Calculer A et B .
2. Générer, suivant cette loi, une liste d'une dizaine de points répartis entre une pression minimale de 650 mm Hg et une pression maximale de 1050 mm Hg. Calculer d'abord les températures « exactes » suivant la loi $AP + B$, puis bruitez ces données en supposant une incertitude de $\pm 5^\circ\text{C}$ sur les mesures des températures. On supposera négligeables les mesures sur les pressions.
Indication : on utilisera une distribution gaussienne du bruit à partir de la loi normale étudiée plus haut, de valeur moyenne nulle et d'écart type correspondant à l'erreur supposée sur les températures.
3. Tracer sur la même figure, les données expérimentales (fonction `errorbar()` de `matplotlib` et la courbe théorique.
4. On donne un modèle de programme d'écriture des données dans un fichier. Inspirez-vous en pour créer votre fichier de données (pression-température)

```
nomFichier = 'data.txt'
f = open(nomFichier, 'w')
f.write('variable X, variable Y \n')
for k in range(Npt):
    f.write("%5.1f" %X[k])
    f.write("%5.1f \n" %Y[k])
f.close()
```

Indications :

- le caractère `\n` indique un retour à la ligne
- l'écriture `"%5.1f" %a` indique que la variable a est un réel (caractère `f`) affiché avec 5 chiffres significatifs et 1 seul chiffre derrière la virgule.

B Ajustement de courbes

On se propose dans cette partie de récupérer les données stockées dans un fichier (le fichier précédent fera l'affaire), puis de les traiter afin d'extraire les paramètres

1. une pression de 1 bar correspond à 760 mm Hg

d'ajustement correspondant à une modélisation du phénomène physique étudié.

On donne tout d'abord un modèle de fichier de récupération des données :

```
# -*- coding: cp1252 -*-
"""
*****
Lecture d'un fichier de données "expérimentales"
*****
"""

from __future__ import division
from scipy import *
from pylab import *

f = open("data.txt", 'r')
f.readline() # saute la première ligne
x_tab = array([])
y_tab = array([])
for ligne in f:
    mots = ligne.split()
    x = float(mots[0])
    y = float(mots[1])
    x_tab = append(x, x_tab)
    y_tab = append(y, y_tab)

print x_tab, y_tab
plot(x_tab, y_tab, 'o')
show()
```

Vérifier que vous pouvez récupérer vos données avec ce programme.

B.1 Ajustement linéaire

Méthode des moindres carrés

Le principe de cette méthode consiste à déterminer la « meilleure droite » $y(x)$ passant par les points expérimentaux y_i en minimisant la distance de cette droite $Ax + B$ aux différents points mesurés y_i . Pour cela on construit la fonction

$$\chi^2(A, B) = \sum_{i=1}^N [y_i - (Ax_i + B)]^2$$

qui est extrémale si $\frac{\partial \chi^2}{\partial A} = \frac{\partial \chi^2}{\partial B} = 0$.

On obtient (faites le calcul chez vous) le système linéaire en A et B

$$\begin{cases} A \sum x_i^2 + B \sum x_i &= \sum x_i y_i \\ A \sum x_i + B N &= \sum y_i \end{cases}$$

Ce qui donne

$$A = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{\Delta} \quad \text{et} \quad B = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{\Delta}$$

avec $\Delta = N \sum x_i^2 - (\sum x_i)^2$.

On suppose que tous les points y_i sont soumis à la même loi de distribution dont on estime l'écart type à l'aide de l'expression

$$\sigma_y^2 = \frac{1}{N-2} \sum_{i=1}^N [y_i - (Ax_i + B)]^2$$

et en utilisant la formule de propagation des erreurs, on en déduit l'incertitude sur A et B

$$\sigma_A = \sigma_y \sqrt{\frac{N}{\Delta}} \quad \text{et} \quad \sigma_B = \sigma_y \sqrt{\frac{\sum x_i^2}{\Delta}}$$

Exercice n° 6 Programme « maison »

Traiter les données en appliquant les formules précédentes de la méthode des moindres carrés. Calculer A , B , σ_A et σ_B et comparer avec les valeurs « exactes » qui nous ont permis de construire les données de la simulation.

Exercice n° 7 Programme scipy

À l'aide de la fonction `curve_fit` de la bibliothèque `scipy.optimize` on peut traiter directement les données en définissant la fonction `fonc(x, a, b)` à ajuster. On récupère les paramètres A et B optimisés de la façon suivante :

```
def fonc(x, a, b):
    return a*x+b

popt, pcov = curve_fit(fonc, P, T)
[A, B] = popt

sigma_A = sqrt(pcov[0, 0])
sigma_B = sqrt(pcov[1, 1])
```

où `popt` est un vecteur qui renvoie les paramètres optimisés de la fonction et `pcov` est une matrice dont les éléments diagonaux correspondent aux écarts types aux carrés (variances) de chacun des paramètres. Comparer avec les résultats de votre programme précédent.

B.2 Ajustement non linéaires

Exercice n° 8 *Sinusoïde amortie*

On reprend la démarche suivie précédemment mais en supposant cette fois-ci l'évolution temporelle d'une tension suivant une sinusoïde amortie de la forme

$$V(t) = Ae^{-t/\tau} \cos(2\pi f t + \phi)$$

avec par exemple $A = 2$, $\tau = 10$, $f = 1$ et $\phi = -\frac{\pi}{5}$

1. Écrire un premier programme qui crée un fichier de données « bruitées » échantillonnant environ 200 points sur l'intervalle de temps $[0, 20]$.
2. Écrire ensuite un second programme qui relit le fichier précédent et recherche les paramètres optimisés ainsi que leurs incertitudes. Commentez.

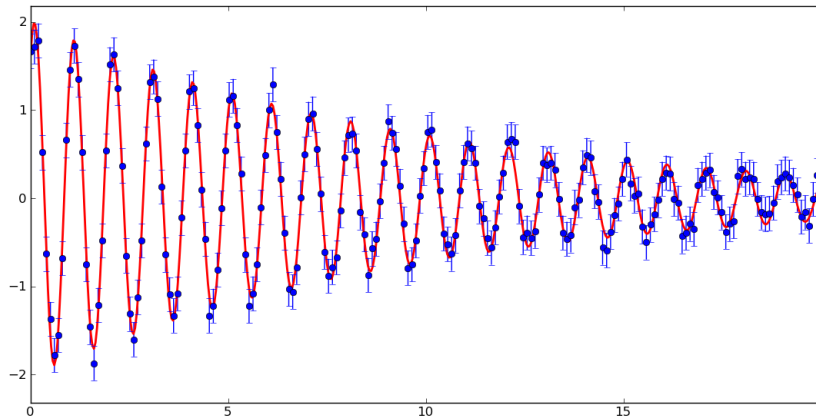


FIGURE 3 – *Meilleur ajustement d'une sinusoïde amortie*