Travaux numériques 2 – Régression linéaire

Capacités numériques

- simuler, à l'aide d'un langage de programmation ou d'un tableur, un processus aléatoire de variation des valeurs expérimentales de l'une des grandeurs – simulation Monte-Carlo – pour évaluer l'incertitude sur les paramètres du modèle ;
- utiliser les fonctions de base des bibliothèques `random` et/ou `numpy` (leurs spécifications étant fournies) pour réaliser des tirages d'une variable aléatoire ;
- utiliser la fonction `hist` de la bibliothèque `matplotlib`.pyplot (sa spécification étant fournie) pour représenter les résultats d'un ensemble de tirages d'une variable aléatoire ;
- utiliser la fonction `polyfit` de la bibliothèque `numpy` (sa spécification étant fournie) pour exploiter des données ;
- utiliser la fonction `random.normal` de la bibliothèque `numpy` (sa spécification étant fournie) pour simuler un processus aléatoire ;
- utiliser les fonctions de base de la bibliothèque `matplotlib` pour tracer la courbe représentative d'une fonction.

Rappels sur les incertitudes

Incertitude-type

Une incertitude-type représente l'écart-type d'une mesure, elle-même considérée comme une variable aléatoire gaussienne. Une mesure comportant une incertitude de type B est une mesure dont on ne sait pas mesurer la variabilité, contrairement à une incertitude de type A. L'incertitude de type B est estimée en fixant un intervalle de largeur ℓ sur lequel la mesure, en tant que variable aléatoire, est supposée être équiprobable : la probabilité que la mesure soit sur l'intervalle $[-\ell/2;\ell/2]$ est constante.

Pour unifier les types d'incertitudes A et B, on ramène l'intervalle d'incertitudes de type B à un écart-type. On peut montrer alors, pour une variable mesurée x :

$$u(x) = \frac{\ell}{2\sqrt{3}}$$

Cela implique pourtant un changement de comportement : la variable modélisée par une distribution gaussienne d'écart-type $u(x)$ peut prendre des valeurs hors de l'intervalle initial (voir figure 1). Mais, c'est le prix à payer pour pouvoir comparer et combiner les incertitudes des différents types. Il faut donc bien utiliser ce modèle de l'incertitude.

Pour illustrer ces explications, j'ai supposé mesurer une distance $x = 1,1$ cm, en estimant les incertitudes de résolution et de pointage à 1 mm de part et d'autres. La simulation de cette mesure est faite grâce au code ci-dessous et la figure 1.

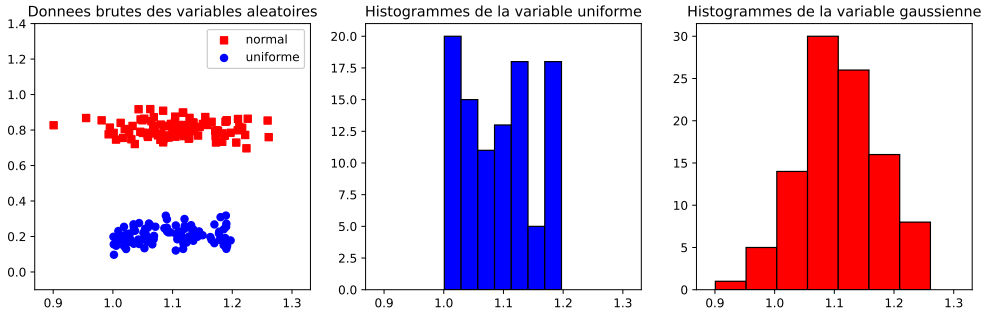

FIGURE 1 – Comparaison des distributions uniformes et gaussiennes.

Exemples de distributions de variables aléatoires *encadré 1*

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

# exemple de mesure avec une incertitude de type B sur [x1;x2]
x1=1.0
x2=1.2
xmoy=(x1+x2)/2. # centre de  [x1;x2]
ux=(x2-x1)/(2.*np.sqrt(3)) # incertitude-type
n=100 # nombre de valeurs
x_normal=rd.normal(xmoy,ux,n) # distribution gaussienne d'ecart-
    type ux
x_uniform=rd.uniform(x1,x2,n) # distribution uniforme sur [x1;x2]
y=rd.normal(0,0.05,n) # tricherie de mise en page

plt.figure(figsize=(14,4)) # taille physique de l'image
ax1=plt.subplot(1,3,1) # pour faire 3 graphes sur la meme figure
ax2=plt.subplot(1,3,2)
ax3=plt.subplot(1,3,3)
ax1.scatter(x_normal,y+0.8,marker='s',c='red',label='normal')
ax1.scatter(x_uniform,y+0.2,marker='o',c='blue',label='uniforme')
ax2.hist(x_uniform,bins=7,histtype = 'bar',edgecolor='black',color
    ='blue')
ax3.hist(x_normal,bins=7,histtype = 'bar',edgecolor='black',color='
    red')
ax1.set_title("Donnees brutes des variables aleatoires")
ax2.set_title("Histogrammes de la variable uniforme")
ax3.set_title("Histogrammes de la variable gaussienne")
ax1.set_ylim(-0.1,1.4)
ax1.set_xlim(xmoy-4*ux,xmoy+4*ux)
ax2.set_xlim(xmoy-4*ux,xmoy+4*ux)
ax3.set_xlim(xmoy-4*ux,xmoy+4*ux)
ax1.legend()
plt.savefig("distributions.eps") # extension a changer au besoin
# plt.show()
```

Simulation Monte-Carlo

La simulation numérique de mesures, et donc la prise en compte d'incertitudes par une (ou plusieurs) variable aléatoire est très courante en physique contemporaine. On veut, par exemple en physique des particules, savoir si un détecteur permettra de mesurer un signal faible (le passage de quelques particules rares) malgré les incertitudes et le reconnaître malgré tous les évènements aléatoires possibles. Cette technique s'appelle une simulation *Monte-Carlo*.

Ici, nous voulons mettre en œuvre une simulation Monte-Carlo pour vérifier un lien linéaire entre deux variables mesurées et connaître les paramètres du modèle.

Ajustements

introduction

Dans une étude expérimentale, le premier temps consiste à reconnaître les processus à l'œuvre. Le deuxième temps vise à mettre en lumière les processus liés par une corrélation, ou par un lien de cause à effet, grâce à la mesure de plusieurs grandeurs. Enfin, s'il y a un lien entre deux grandeurs, on cherche à lui donner une forme mathématique.

Visuellement, il est possible de reconnaître les variables aléatoires indépendantes (figure 2-(a)), les relations linéaires (fig. 2-(b)), et c'est à peu près tout. Certains disent qu'il nous est particulièrement facile de détecter une relation linéaire. En tous cas, on reconnaît la présence d'une courbure, comme dans la fig. 2-(c). Pour déterminer la forme précise d'un lien non-linéaire, c'est difficile.

Concentrons-nous sur un lien linéaire entre deux grandeurs x et y, en partie aléatoires. On veut déterminer la forme exacte de ce lien, pour cela on utilise la méthode des moindres carrés.

Méthode des moindres carrés

Soit une variable x, mesurée avec une incertitude $u(x)$. x est le paramètre d'une expérience dont le résultat est une variable y mesurée avec une incertitude $u(y)$. Supposons que l'on s'attende à avoir le lien théorique :

$$y = ax + b$$

avec a et b deux grandeurs physiques.

Supposons avoir mesuré un ensemble de N couples de valeurs (x_i, y_i). En raisonnant un peu rapidement, on s'attendrait à trouver $y_i = ax_i + b$, avec a et b fixés. Ce n'est pas le cas car les x_i et y_i sont des valeurs aléatoires. En revanche, on peut construire une valeur $\tilde{y}_i = ax_i + b$, et comparer les valeurs y_i et $\tilde{y}_i$. Le but est de trouver les valeurs $\tilde{y}_i$ les plus proches des y_i lorsque l'on fixe les valeurs de a et b. Pour cela, on construit la fonction $S(a,b)$ à partir de toutes les différences $\tilde{y}_i - y_i$ et on minimise cette fonction. La somme des différences est forcément proche de zéro, puisque les points doivent être de part et d'autres de la droite.

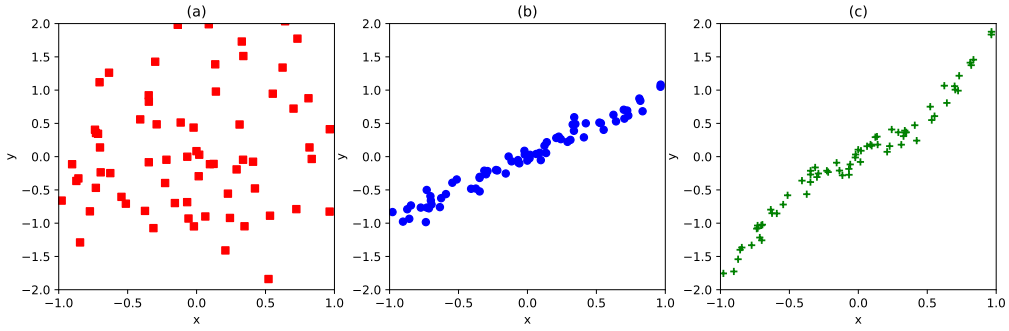

FIGURE 2 – Illustration de lien entre des grandeurs en partie aléatoire. (a) x et y sont aléatoires, gaussiennes et indépendantes ; (b) x est gaussienne et $y = x + \varepsilon$ avec ε est une variable aléatoire gaussienne ; (c) *idem* (b) avec $y = x^3 + x + \varepsilon$.

Donc on construit la somme de termes tous positifs, en prenant le carré de chaque terme, d'où le nom de la méthode :

$$S(a,b) = \sum_{i=1}^{N} \left(\tilde{y}_i - y_i\right)^2 = \sum_{i=1}^{N} \left(ax_i + b - y_i\right)^2$$

$$\frac{\partial S}{\partial b} = 0 = 2\sum_{i=1}^{N} \left(ax_i + b - y_i\right)$$

$$\Rightarrow b = \frac{1}{N}\sum_{i=1}^{N} y_i - a\frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{ou} \quad \boxed{b = < y > - a < x >}$$

$$\frac{\partial S}{\partial a} = 0 = 2\sum_{i=1}^{N} x_i \left(ax_i + b - y_i\right) = 2\sum_{i=1}^{N} x_i \left(ax_i + b - y_i\right) - 2 < x > \sum_{i=1}^{N} \left(ax_i + b - y_i\right)$$

$$= 2\sum_{i=1}^{N} \left[x_i - < x >\right] \left[a(x_i - < x >) - (y_i - < y >)\right]$$

$$\Rightarrow a = \frac{\sum_{i=1}^{N} \left[x_i - < x >\right] \left[y_i - < y >\right]}{\sum_{i=1}^{N} \left[x_i - < x >\right]^2} \quad \text{ou} \quad \boxed{a = \frac{\mathrm{Cov}(x_i, y_i)}{\mathrm{Var}(x_i)}}$$

Remarque : cette méthode ne fait pas partie du programme de physique. Mais c'est un exercice classique, et cela montre que la recherche de la droite d'ajustement provient d'une minimisation. La plupart des logiciels contiennent un élément permettant d'obtenir cette droite. Avec `Python`, il y en a plusieurs en fonction des bibliothèques utilisées, par exemple :

```
numpy.polynomial.polynomial.polyfit(x, y, deg)
```

La méthode présentée est la plus simple des méthodes des moindres carrés et elle ne tient pas compte de l'incertitude de chacun des couples (x_i, y_i).

Prise en compte des incertitudes

En fait chacune des valeurs x_i (et *idem* pour y_i) est un évènement aléatoire, dont on connaît la distribution, gaussienne, et l'écart-type $u(x)$ (et $u(y)$). C'est aléatoire, donc un

deuxième expérimentateur, sur la même expérience, aurait pu avoir un autre jeu de valeurs (x_i, y_i), et aussi une autre droite définie par une autre pente a_2 et une autre ordonnée à l'origine b_2. Un troixième expérimentateur aura encore un autre résultat, *etc.*

Si on fait un grand nombre de fois l'expérience, on obtiendra des jeux (a_j, b_j) dont on pourra faire une étude statistique, calculer moyenne et écart-type pour obtenir $a \pm u(a)$ et $b \pm u(b)$. Mais, connaissant les propriétés statistiques des x_i et des y_i, au lieu de refaire l'expérience, on peut en faire une simulation par la méthode de Monte-Carlo et atteindre le même résultat.

Pour résumer la procédure :

1. on crée un jeu de valeurs (x_i, y_i) par un générateur de nombres aléatoires gaussiens et d'écarts-types $(u(x_i), u(y_i))$;
2. on détermine la droite ajustant les couples (x_i, y_i), on en déduit un couple (a_1, b_1) :
3. on recommence n fois les point 1 et 2 pour obtenir l'ensemble des (a_j, b_j) avec $j = 1, ..., n$, où n est fixé ;
4. on étudie la statistique des (a_j, b_j).

Travail à réaliser

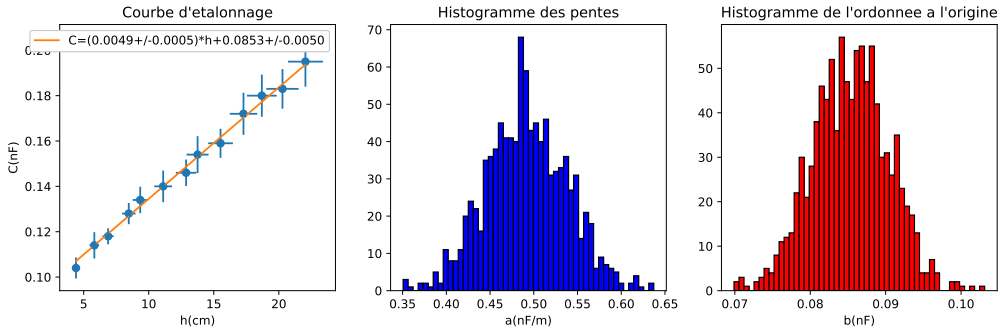

FIGURE 3 – Travail demandé.

On reprend le TP 7 : une éprouvette contient un capteur capacitif de niveau d'eau en série duquel on branche un générateur de fonctions, et une résistance AOIP $R = 5,0.10^5\,\Omega$ à 1% près. On veut faire la droite d'étalonnage de ce capteur. Pour cela, on ajoute de l'eau dans l'éprouvette graduée, on lit le volume d'eau. On mesure le temps typique τ grâce à un oscilloscope en se plaçant à 63% de la charge du condensateur. En fait, on fait deux mesures de temps, encadrant τ par valeur inférieure τ_1 et par valeur supérieure τ_2.

On obtient alors la série de mesures :

V (mℓ)	50	66	78	96	106	126	146	156	176	196	212	230	250
τ_1 (μs)	48	52	56	60	62	64	68	70	74	78	82	84	88
τ_2 (μs)	56	62	62	68	72	76	78	84	85	94	98	99	107

On estime l'incertitude de lecture et de pointage du volume par un intervalle de largeur de 4 mℓ. L'éprouvette a un diamètre entre $d = 3,6$ cm et $4,0$ cm.

Vous ferez un unique programme `Python` permettant de (voir figure 3) :

1. tracer les points expérimentaux de la courbe d'étalonnage de la capacité en fonction de la hauteur d'eau,
2. déterminer l'ajustement linéaire modélisant la courbe précédente en utilisant la méthode de Monte-Carlo pour estimer les incertitudes types des paramètres de la droite (votre programme affichera ces valeurs),
3. tracer la droite précédement déterminée sur le graphique des points expérimentaux,
4. tracer les histogrammes des pentes des droites obtenues lors de la méthode de Monte-Carlo,
5. *idem* pour les ordonnées à l'origine.

Listes des fonctions utilisables, en plus de celles données dans le programme précédent :

— `numpy.mean(x)` : moyenne de x
— `numpy.std(x, ddof=1)` : écart-type de x
— `matplotlib.pyplot.errorbar(x, y, uy, ux)` : barres d'erreur des points
— `coef=numpy.polynomial.polynomial.polyfit(x, y, deg)` : le tableau `coef` contient les coefficients du polynôme ajustant les points (x, y) par ordre croissant de puissance du polynoôme.
— `label="(:.4f+/-:.4f)*h".format(a, b)` : mise en format de deux valeurs, prises dans l'ordre `a` puis `b`, sous la forme de deux réels à virgule avec quatre décimales.