

Physique

Fiche – Régression linéaire et méthode de Monte-Carlo

L. TORTEROTOT

1 Régression linéaire

Principe Selon le modèle physique considéré, éventuellement sous certaines conditions, il peut exister une relation linéaire¹ entre deux grandeurs physiques x et y , telle que $y = ax + b$.

✎ Les grandeurs y et x ne sont pas nécessairement accessibles par une mesure directe. La relation de Snell-Descartes est par exemple linéaire entre les sinus des angles d'incidence et de réfraction.

La détermination des paramètres a et b peut se faire expérimentalement en mesurant les valeurs de y pour des valeurs de x données. Alors, il est possible de placer les points correspondant à ces mesures dans le plan (x, y) . Ces points doivent être relativement alignés (il existe toujours une variabilité des mesures). Les paramètres a et b se déterminent alors par la droite *la plus ajustée* à ces points de mesure.

Le choix d'un critère permettant de déterminer *le meilleur* ajustement peut être sujet à discussion, ce n'est pas le propos dans le cadre du programme de CPGE dans lequel le choix fait est d'utiliser `polyfit` de la bibliothèque `numpy`. Alors, avec `x_mes` la liste des valeurs mesurées pour x et `y_mes` la liste des valeurs mesurées pour y , les paramètres a et b de la régression linéaire sont obtenus avec `a, b = np.polyfit(x_mes, y_mes, 1)`.

Cette méthode permet d'obtenir les valeurs de a et b , mais pas leurs incertitudes $u(a)$ et $u(b)$ alors que chaque valeur de x (respectivement, de y) dans `x_mes` (respectivement `y_mes`) possède une incertitude. La méthode de Monte-Carlo permet de déterminer $u(a)$ et $u(b)$.

Exemple concret Détermination d'un indice optique à partir de :

- mesures de l'angle incident avec un rapporteur gradué au degré et $y = \sin \theta_i$;
- mesures de l'angle réfracté avec un rapporteur gradué au degré et $x = \sin \theta_r$;
- la relation de Snell-Descartes.

Comme $\sin \theta_i = n \sin \theta_r$, $a = n$ et $b = 0$ *a priori*. Les mesures des angles θ_i et θ_r permettent de placer les points de la figure 1 et `polyfit` donne a et b (permettant de tracer la droite de régression), mais sans leurs incertitudes. Si les points semblent bien s'aligner sur une droite, impossible en revanche de dire :

- si la valeur de a est cohérente avec la valeur attendue pour le milieu étudié, en l'occurrence $4/3$, car $1,34 \neq 4/3$, mais l'écart est-il significatif ?
- si la valeur de b est cohérente avec la valeur attendue d'après le modèle, en l'occurrence 0 , car $-3,1 \times 10^{-3} \neq 0$, mais l'écart est-il significatif ?

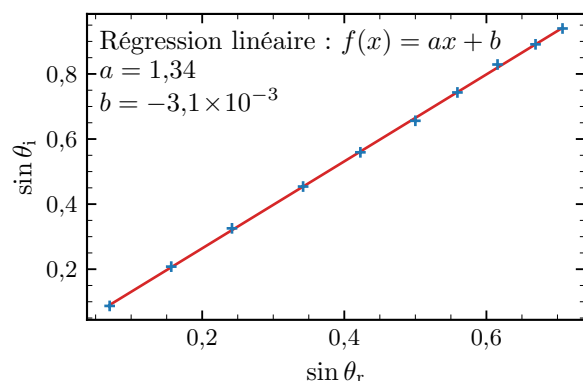


Figure 1 – Régression linéaire, sans incertitudes.

2 Estimation d'une incertitude-type avec la méthode de Monte-Carlo

2.1 Cas d'une mesure unique, remplacement de la formule de propagation

Position du problème Lorsqu'une grandeur y s'exprime en fonction de M autres grandeurs mesurées $x_m, m \in \{1, \dots, M\}$ d'incertitudes types $u(x_m)$ connues, il est parfois possible de déterminer par le calcul l'incertitude type sur y , $u(y)$.

1. Au sens physique du terme, ce qui correspond à « affine » en mathématiques.

En effet, lorsque les grandeurs d'entrée x_m sont *indépendantes*, l'incertitude-type composée de y est obtenue par la formule de propagation des incertitudes-type dont découlent les deux formules à connaître en CPGE :

Cas d'une somme $y = ax_1 + bx_2$

$$u(y) = \sqrt{(a u(x_1))^2 + (b u(x_2))^2}.$$

Cas d'un produit $y = x_1^a x_2^b$

$$\frac{u(y)}{y} = \sqrt{\left(\frac{a u(x_1)}{x_1}\right)^2 + \left(\frac{b u(x_2)}{x_2}\right)^2}.$$

Dans certains cas, ces formules sont suffisantes². En revanche, si les incertitudes sur les x_m ne sont pas indépendantes ou si l'expression de y n'est pas simple, le calcul se complique rapidement. Une simulation permet toutefois d'obtenir une (bonne) estimation de $u(y)$.

Principe La méthode de Monte-Carlo consiste à simuler $N \gg 1$ fois le calcul de y à partir des mesures de x_m . Si $y = f(x_m)$, alors à la n^e itération, la valeur simulée de y est $y_n = f(x_{m,n})$ avec $x_{m,n}$ tirés aléatoirement autour des x_m selon les distributions de probabilité associées aux x_m . Alors, une liste $\{y_n\}$ de N valeurs simulées de y est obtenue. L'incertitude type sur y est alors estimée comme étant l'écart-type de cette liste, car y pourrait être une valeur de cette liste (et non sa moyenne).

Exemple concret Détermination d'un indice optique à partir de :

- la mesure de l'angle incident avec un rapporteur gradué au degré $\theta_i = (45,00 \pm 0,29)^\circ$ ³ ;
- la mesure de l'angle réfracté avec un rapporteur gradué au degré $\theta_r = (32,00 \pm 0,29)^\circ$;
- la relation de Snell-Descartes, $n = \sin(\theta_i)/\sin(\theta_r)$.

Le calcul de n est facile ($n \simeq 1,3344\dots$), mais pas celui de $u(n)$. Or, θ_i (comme θ_r) se trouve nécessairement au plus à Δ_{tol} (la tolérance) autour de la valeur mesurée, avec une distribution de probabilité uniforme⁴, la tolérance étant ici égale à une demie graduation (sinon, la mesure aurait donné la graduation à côté), c'est-à-dire $\Delta_{\text{tol}} = \frac{1}{2}\Delta_{\text{grad}} = 0,5^\circ$.

Le code ci-après permet de réaliser cette démarche avec les notations se correspondant conformément au tableau 1. Les distributions des variables aléatoires sont illustrées en figure 2.

Alors, $u(n) = 0,012712\dots$ donc⁵ $n = (1,334 \pm 0,013)$.

```
6 import numpy as np
7 import random as rd
8
9 theta_i_deg = 45.00 # valeur mesurée
10 theta_r_deg = 32.00 # valeur mesurée
11 tolerance_deg = 1/2 # tolérance d'une demie graduation
12 # calcul de n à partir des valeurs mesurées :
13 ind_opt = np.sin(theta_i_deg * np.pi/180)/np.sin(theta_r_deg * np.pi/180)
14
15 N = 10000 # nombre d'itérations pour Monte-Carlo
16 inds_opt = [] # initialisation de la liste des valeurs de l'indice
17 for n in range(N):
18     theta_i_deg_n = rd.uniform(theta_i_deg - tolerance_deg, theta_i_deg +
19     tolerance_deg) # tirage d'une valeur de theta_i aléatoirement, loi uniforme
```

Générale	Exemple	Code
x_1	θ_i	theta_i_deg
x_2	θ_r	theta_r_deg
y	n	ind_opt
$x_{1,n}$	-	theta_i_deg_n
$x_{2,n}$	-	theta_r_deg_n
y_n	-	ind_opt_n
N	-	N

Tableau 1 – Correspondance des notations.

2. Quitte, si par exemple $y = x_1 + x_2/x_3$, à décomposer en sommes et produits pour appliquer les formules exigibles.

3. La détermination de $u(\theta_i)$ est l'objet de la fiche « Incertitudes expérimentales ».

4. Pourquoi la valeur réelle pour une mesure sur des graduations serait-elle plus au centre qu'aux bords de l'intervalle ?

5. Voir la fiche « Incertitudes expérimentales » pour les règles d'écriture du résultat.

```

19  theta_r_deg_n = rd.uniform(theta_r_deg - tolerance_deg, theta_r_deg +
    ↪  tolerance_deg) # tirage d'une valeur de theta_r aléatoirement, loi uniforme
20  theta_i_rad_n = theta_i_deg_n * np.pi/180 # conversion en rad
21  theta_r_rad_n = theta_r_deg_n * np.pi/180 # conversion en rad
22  ind_opt_n = np.sin(theta_i_rad_n)/np.sin(theta_r_rad_n) # Snell-Descartes
23  inds_opt.append(ind_opt_n)
24
25  u_ind_opt = np.std(inds_opt, ddof = 1) # obtenir l'incertitude type
26  # affichage des résultats :
27  print(f"u(n) = {np.round(u_ind_opt, 2-int(np.log10(abs(u_ind_opt))))}")
28  print(f"donc n = {np.round(ind_opt, 2-int(np.log10(abs(u_ind_opt))))}")

```

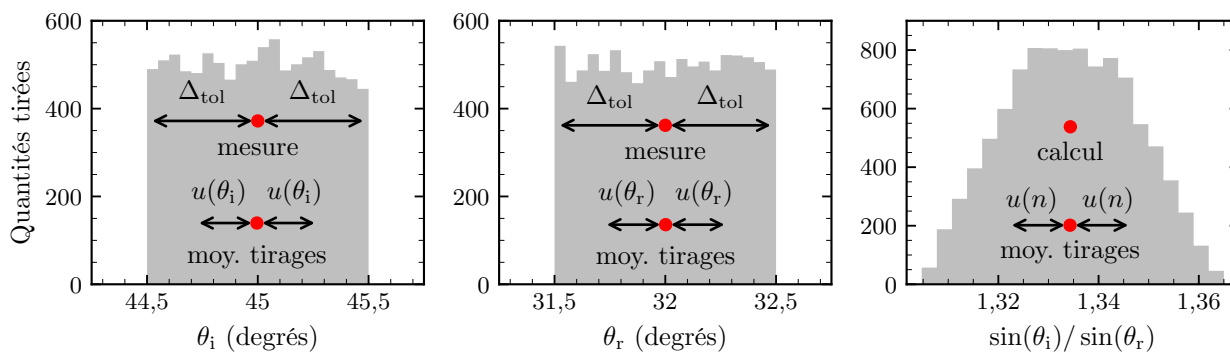


Figure 2 – Histogrammes des valeurs obtenues avec le code Python.

2.2 Cas des paramètres a et b d'une régression linéaire

Principe Le principe reste le même, simuler $N \gg 1$ fois l'expérience et effectuer la régression sur les points de mesures simulés. À la n^e simulation, pour chaque point de mesure m , il faut déterminer le couple $(x_{m,n}, y_{m,n})$. Pour cela, $x_{m,n}$ est tiré aléatoirement dans $x_{m,mes} \pm \Delta_x$ avec Δ_x la tolérance sur x . De même, $y_{m,n}$ est tiré aléatoirement dans $y_{m,mes} \pm \Delta_y$. Alors, N valeurs de a et de b sont obtenues et $u(a)$ et $u(b)$ sont estimées comme étant les écarts-types de ces listes de valeurs. Le code ci-après permet de réaliser cette démarche avec x_mes la liste des valeurs mesurées pour x et y_mes celle pour y .

```

17  a, b = np.polyfit(x_mes, y_mes, 1) # valeurs de a et b avec polyfit
18
19  valeurs_a, valeurs_b = [], [] # initialisation des listes
20  for n in range(N):
21      x_n, y_n = [], []
22      for m in range(len(x_mes)):
23          x_n.append(
24              rd.uniform(x_mes[m] - Delta_x, x_mes[m] + Delta_x)
25              ) # tirage aléatoire d'une valeur pour x_mn
26          y_n.append(
27              rd.uniform(y_mes[m] - Delta_y, y_mes[m] + Delta_y)
28              ) # tirage aléatoire d'une valeur pour y_mn
29      a_n, b_n = np.polyfit(x_n, y_n, 1) # valeurs des paramètres obtenus
30      valeurs_a.append(a_n) # lors de cette itération et
31      valeurs_b.append(b_n) # leur stockage dans les listes
32
33  u_a = np.std(valeurs_a, ddof = 1) # incertitude-type = écart-type
34  u_b = np.std(valeurs_b, ddof = 1)

```

Quelques remarques

- Les tolérances ne sont bien évidemment pas nécessairement les mêmes pour x et y ni pour les différents points de mesure ;
- Les valeurs de a et b sont celles obtenues avec `polyfit` appelé sur les *vraies* mesures (ligne 17 du code), ce n'est pas le hasard qui fait la physique !
- Il ne faut pas diviser les écart-types par \sqrt{N} pour avoir $u(a)$ et $u(b)$ car la méthode de Monte-Carlo permet ici d'obtenir l'incertitude sur *une mesure* et non sur *une moyenne*. Sinon, simuler plus (augmenter N) réduirait l'incertitude... alors qu'aucune donnée supplémentaire n'existe !
- Dans l'exemple donné, une loi uniforme sur les intervalles $[x_{\text{mes}} - \Delta_x; x_{\text{mes}} + \Delta_x]$ et $[y_{\text{mes}} - \Delta_y; y_{\text{mes}} + \Delta_y]$ est utilisée car adaptée au cas de mesures sur des graduations. Dans le cas général, réfléchir à comment varie chaque paramètre et ne pas hésiter à laisser à Python le soin d'appliquer des formules de propagation.


Exemple concret Sur la figure 3, les points placés sont les mêmes que sur la figure 1 mais les barres d'incertitude sont ajoutées (elles sont relativement petites). Le code présenté précédemment, appliqué au cas présent, permet d'obtenir $n = (1,3361 \pm 0,0054)$, ce qui est plus précis que dans l'exemple de la mesure unique. Une régression linéaire est plus précise qu'une mesure unique car en elle comporte plusieurs.

De plus, l'écart normalisé de b à la valeur nulle est $|b|/u(b)$ et est inférieur à 2. Il est désormais possible de dire que la valeur non nulle du paramètre b obtenue par régression linéaire est compatible avec 0 et ne remet donc pas en cause le modèle.

Enfin, l'écart normalisé de $n = a$ à la valeur attendu $n_{\text{th}} = \frac{4}{3}$ est

$$E_N = \frac{|n - n_{\text{th}}|}{\sqrt{u(n)^2 + u(n_{\text{th}})^2}} \quad \text{soit ici avec } u(n_{\text{th}}) = 0 \quad E_N = \frac{|n - n_{\text{th}}|}{u(n)} = 0,512 < 2,$$

la valeur expérimentale est donc cohérente avec la valeur théorique.

 Avec une même valeur « centrale » de n , si l'incertitude avait été 10 fois plus faible, l'écart normalisé aurait été plus grand que 2 malgré un écart absolu identique. C'est donc bien avec une détermination de l'incertitude associée à une mesure, et non avec seulement une mesure, qu'il est possible de trancher sur la validité ou non d'un résultat ou d'un modèle.

« La notion d'incertitude est essentielle dans la démarche expérimentale. Sans elle, on ne peut juger de la qualité d'une mesure, de sa pertinence ou de sa compatibilité avec une loi physique. » [1]

Références

- [1] F.-X. BALLY & J.-M. BERROIR. « Incertitudes expérimentales ». *BUP* **928** (nov. 2010).

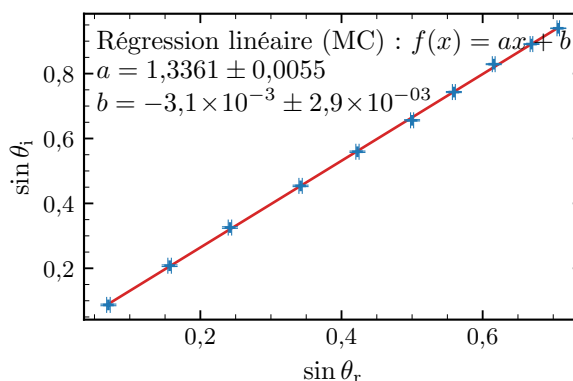


Figure 3 – Régression linéaire, avec incertitudes et méthode de Monte-Carlo.