Jean-Noël Beury

PHYSIQUE

PSI/PSI*

EXERCICES INCONTOURNABLES

j'intègre

3e édition

DUNOD

Avec la collaboration scientifique de SÉBASTIEN FAYOLLE

Couverture : création Hokus Pokus, adaptation Studio Dunod

11 rue Paul Bert, 92240 Malakoff
www.dunod.com

ISBN 978-2-10-084982-6

Table des matières

Partie 5
Électromagnétisme

Partie 6
Conversion de puissance

Partie 7
Ondes

Les énoncés dans lesquels apparaît un astérisque annoncent des exercices plus difficiles.

Vous pouvez télécharger les programmes Python des exercices à partir de la page de présentation de l'ouvrage sur le site Dunod.

https://dunod.com/EAN/9782100849826

Partie 1

Mesures et incertitudes

Plan

1 Capacités numériques

Exercice 1.1 : Incertitudes - Évaluation de type A

On cherche à mesurer la distance focale image d'une lentille convergente avec la méthode d'autocollimation. On réalise 6 observations : 20,1 cm ; 19,1 cm ; 19,9 cm ; 20,1 cm ; 20,3 cm et 19,7 cm.

```
import numpy as np   #bibliothèque numpy renommée np
np.std(data, ddof=1) #renvoie l'écart-type du tableau data avec N-1
                     #au dénominateur pour un tableau contenant
                     #N valeurs
```

1. Écrire le programme Python permettant de calculer l'incertitude-type associée à une unique observation.

2. Pour mesurer au mieux la distance focale, on calcule la valeur mesurée à l'aide d'une moyenne arithmétique. Écrire le programme Python permettant de calculer la moyenne des observations et l'incertitude-type associée à cette moyenne.

3. Écrire, avec un nombre adapté de chiffres significatifs, le résultat de la mesure de la distance focale image de la lentille.

Analyse du problème

La mesure est intrinsèquement variable. On distingue l'évaluation de type A de l'évaluation de type B. Pour l'évaluation de type A, il faut bien distinguer l'incertitude-type associée à une observation unique de l'incertitude-type associée à la moyenne des N observations.

Cours :

La grandeur que l'on souhaite mesurer s'appelle le mesurande. Le résultat de la mesure est un ensemble de valeurs numériques raisonnablement attribuables à la grandeur d'intérêt. La dispersion de cet ensemble est évaluée en utilisant l'incertitude-type. On écrit l'incertitude-type avec deux chiffres significatifs. On arrondit au plus près l'incertitude-type. On rencontre deux cas :

- **Mesure répétée plusieurs fois : Évaluation de type A de l'incertitude**

  On restreint l'ensemble des observations x_i à trois paramètres :

  – Moyenne arithmétique des observations : $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

```
import numpy as np  #bibliothèque numpy renommée np
np.mean(x)          #calcul de la moyenne du tableau x ou de la
                    #liste x
```

– Incertitude-type sur une mesure : $u(x) = \sqrt{\dfrac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

```
import numpy as np
np.std(x, ddof=1)   #calcul de l'écart-type de x avec N-1 au
                    #dénominateur
```

On peut montrer que le meilleur estimateur de l'écart-type de la variable aléatoire x n'est pas $\sqrt{\dfrac{1}{N}\sum_{i=1}^{n}(x_i - \bar{x})^2}$ mais $\sqrt{\dfrac{1}{N-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$. Avec Python, on utilise `np.std(x, ddof=1)` pour calculer l'écart-type avec $N-1$ au dénominateur.

– Incertitude-type sur la moyenne $\bar{x}$: l'écart-type de la moyenne (qui est également une variable aléatoire) ou l'incertitude-type sur la moyenne vaut : $u(\bar{x}) = \dfrac{u(x)}{\sqrt{N}}$.

Avec Python, on utilise :

```
np.std(x, ddof=1)/np.sqrt(len(x))
```

• **Mesure réalisée une seule fois : Évaluation de type B de l'incertitude**

Si on a peu d'informations sur le mesurande, on fait l'hypothèse raisonnable que le résultat de la mesure est décrit par une variable aléatoire suivant une loi uniforme entre deux bornes : $[m - \Delta; m + \Delta]$. On appelle Δ la demi-étendue (ou demi-largeur) de l'intervalle. L'incertitude-type vaut : $u = \dfrac{\Delta}{\sqrt{3}}$.

– Demi-graduation pour la lecture d'une distance avec une règle graduée : Δ = 0,5 mm. On peut prendre également Δ = 1 mm si on n'est pas sûr de la lecture.

– Voltmètre : le constructeur donne souvent la précision. Par exemple pour la mesure de tension DC, la précision est : 0,2 % de la lecture + 1 UL. On appelle UL ou UR l'unité de lecture ou de représentation. Tous les digits de l'afficheur sont à 0 sauf le dernier à 1. Pour une lecture U = 5,10 V (calibre 40 V), un digit représente 10 mV. La précision ou la demi-étendue vaut $\Delta = \dfrac{0{,}2}{100} \times 5{,}10 + 0{,}01$. L'incertitude-type est : $u(V) = \dfrac{\Delta}{\sqrt{3}} = \dfrac{\frac{0{,}2}{100} \times 5{,}10 + 0{,}01}{\sqrt{3}} = 11{,}7$ mV.

1.

```
import numpy as np       #bibliothèque numpy renommée np
data=[20.1, 19.1, 19.9, 20.1, 20.3, 19.7] #liste des distances
                                          #focales
N=len(data)  #nombre d'observations
moy=np.mean(data)        #moyenne des valeurs
                         #de la liste data
sx=np.std(data, ddof=1)  #écart-type de la liste data avec
                         #N-1 au dénominateur
print('moyenne arithmétique = ', moy)
print('incertitude-type associée à une unique ',\
   'observation = ', sx)
```

L'incertitude-type associée à une unique observation vaut $u(f') = 0{,}43\,\text{cm}$. Elle quantifie la variabilité inhérente au processus d'observation.

2. Pour calculer la distance focale image de la lentille, on prend la moyenne des 6 observations.

- L'écart-type expérimental $u(f') = 0{,}43\,\text{cm}$ indique la variabilité type de la mesure d'une distance focale.
- L'incertitude-type associée à f' est l'incertitude d'une moyenne des observations.

```
print('incertitude-type associée à la moyenne = ',\
   sx/np.sqrt(N))
```

3. On utilise deux chiffres significatifs pour l'incertitude-type. Il y a plusieurs écritures possibles du résultat de la moyenne de la distance focale image :

- $f' = 19{,}87\,\text{cm}$ et $u\left(\bar{f}'\right) = 0{,}17\,\text{cm}$;
- $f' = 19{,}87 \pm 0{,}17\,\text{cm}$ (ce qui suit $\pm$ représente l'incertitude-type).

Exercice 1.2 : Variabilité d'une grandeur composée

On mesure avec un voltmètre U = 44,1 V la tension aux bornes d'une résistance. On mesure avec un ampèremètre I = 0,09 A l'intensité qui la traverse. La température extérieure T_e = 22 °C est mesurée avec un capteur de température. On mesure avec une règle au mm près la longueur L = 5,0 cm et la largeur ℓ = 2,8 cm de la résistance de forme rectangulaire. La surface d'échange avec l'air vaut $S = L\ell$. On enregistre la température T de la résistance en fonction du temps. Par une modélisation, on obtient la température T_p = 36,3 °C avec une incertitude-type égale à 0,1 °C.

Multimètre

Measurement	Range	Resolution	Accuracy ± ([% of reading] + [counts])
DC millivolts	600.0 mV	0.1 mV	0.5 % + 2
DC volts	6.000 V	0.001 V	0.5 % + 2
DC volts	60.00 V	0.01 V	0.5 % + 2
DC volts	600.0 V	0.1 V	0.5 % + 2
DC amps	6.000 A	0.001 A	1.0 % + 3
DC amps	10.00 A	0.01 A	1.0 % + 3

Capteur de température

Fonction	Etendue de mesure	Résolution	Précision
Sonde de température	-25 °C à +125 °C	0,15 °C	0,5 °C

```
import numpy.random as rd  #module numpy.random renommé rd
rd.random() #renvoie un nombre aléatoire suivant une loi
   #uniforme dans la plage semi-ouverte [0.0, 1.0[
rd.uniform(borne_inf, borne_sup) #renvoie un nombre aléatoire
   #suivant une loi uniforme (ou rectangulaire)
   #entre borne_inf et borne_sup
rd.normal(valeur_centrale, incertitude_type #renvoie un nombre
   #aléatoire suivant une loi normale (ou gaussienne)
   #centrée sur valeur_centrale et d'écart-type incertitude_type
rd.uniform(borne_inf, borne_sup, N) #renvoie un tableau contenant
   #N variables aléatoires suivant une loi uniforme
   #(ou rectangulaire) entre borne_inf et borne_sup
rd.normal(valeur_centrale, incertitude_type, N)#renvoie un tableau
   #contenant N variables aléatoires suivant une loi normale
   #(ou gaussienne) centrée sur valeur_centrale et
   #d'écart-type incertitude_type
import numpy as np   #bibliothèque numpy renommée np
np.std(data, ddof=1) #renvoie l'écart-type du tableau data
   #avec N-1 au dénominateur pour un tableau contenant N valeurs
```

1. Calculer les incertitudes-types des grandeurs mesurées U et L. Écrire, avec un nombre adapté de chiffres significatifs, les résultats des mesures de U et L.

2. On cherche à calculer le coefficient de transfert conducto-convectif avec la relation : $h = \dfrac{UI}{(T_P - T_{ext})\,S}$. Utilise-t-on une loi uniforme ou une loi normale pour simuler un processus aléatoire pour les grandeurs U, I, L, ℓ et T_e ? et pour T_P ?

3. Expliquer la méthode de simulation de Monte-Carlo pour calculer l'incertitude-type de h.

4. Écrire un programme Python permettant de calculer la valeur moyenne et l'incertitude-type de h en utilisant des boucles `for`.

5. Écrire un programme Python permettant de calculer la valeur moyenne et l'incertitude-type de h sans utiliser de boucle `for`.

6. Lors de l'exécution du programme Python, on obtient $h = 198{,}35444$ et $u(h) = 7{,}02325$. Écrire, avec un nombre adapté de chiffres significatifs, le résultat de la mesure de h.

Analyse du problème

La méthode de simulation de Monte-Carlo est très pratique à utiliser pour la propagation des incertitudes. On peut ainsi déterminer l'incertitude-type d'une grandeur calculée à partir des grandeurs dont on connaît les incertitudes-types.

Cours :

On se place ici dans le cas où on n'a réalisé qu'une observation unique. Afin de retrouver la variabilité intrinsèque à la mesure, on définit un intervalle dans lequel on est raisonnablement certain que le résultat de la mesure se trouve. Deux possibilités pour retrouver la variabilité de la mesure :

- Si on ne connaît que la limité basse et la limite haute pour la grandeur considérée (par exemple U, I, T_e, L et ℓ), alors on suppose que la **distribution est uniforme** entre ces deux bornes. On utilise une loi uniforme que l'on appelle également rectangulaire.
- Si on connaît l'incertitude-type associée à cette mesure (par exemple T_p), alors on suppose que la **distribution est gaussienne**. On utilise une loi normale dont l'écart-type est l'incertitude-type.

1. En utilisant la notice fournie par le constructeur, la précision vaut $\Delta(U) = \dfrac{0{,}5}{100} \times 44{,}1 + 0{,}02 = 0{,}2405$ V. On peut raisonnablement penser que la tension « réelle » est comprise entre $44{,}1 - \Delta(U)$ et $44{,}1 + \Delta(U)$. La demi-étendue de cet intervalle vaut $\Delta(U)$. Comme on n'a aucune autre information que la limite basse et la limite haute pour la tension, on suppose que la répartition est uniforme entre ces deux bornes. On utilise une loi uniforme ou rectangulaire. L'incertitude-type de la tension vaut $u(U) = \dfrac{\Delta(U)}{\sqrt{3}} = 0{,}14$ V.

On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat de la mesure de U :

- $U = 44{,}10$ V et $u(U) = 0{,}14$ V
- $U = 44{,}10 \pm 0{,}14$ V (ce qui suit $\pm$ représente l'incertitude-type)

Une règle est susceptible de masquer la variabilité associée à la mesure. Tout un ensemble de longueurs aurait pu conduire à 5,0 cm. On peut raisonnablement penser que la longueur « réelle » est comprise entre 4,95 cm et 5,05 cm. On en déduit la demi-étendue de cet intervalle $\Delta(L) = 0{,}5$ cm, soit $u(L) = \dfrac{\Delta(L)}{\sqrt{3}} = 0{,}29$ cm.

On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat de la mesure de L :

- $L = 5{,}00$ cm et $u(L) = 0{,}29$ cm
- $L = 5{,}00 \pm 0{,}29$ cm (ce qui suit $\pm$ représente l'incertitude-type)

Remarques :

- On ne connaît pas les valeurs vraies de la tension U et de la longueur L.
- On a deux zéros derrière la virgule. Dans la valeur mesurée, on perd la notion de chiffres significatifs. On ajoute deux zéros par commodité de lecture. Le deuxième zéro n'a plus de rapport avec la résolution de l'instrument. Il ne faut donc pas lui attribuer de sens physique. L'instrument de mesure ne fixe pas le nombre de chiffres significatifs de la valeur mesurée.

2.

- On ne connaît que les limites basses et hautes pour les grandeurs U, I, L, ℓ et T_e. On utilise une loi uniforme pour retrouver la variabilité des mesures : on simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire ayant les caractéristiques de l'incertitude-type évaluée.

  Par exemple pour U, la valeur mesurée vaut $44{,}1$ V. On simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire comprise entre $-\Delta(U)$ et $\Delta(U)$ suivant une loi uniforme. Avec Python, on peut simuler une variable aléatoire suivant une loi uniforme en utilisant la syntaxe :

  ```
  U + rd.uniform(-delta_U, delta_U)
  ```

  On peut écrire également :

  ```
  U+delta_U*(2*rd.random()-1)
  ```

- On connaît l'incertitude-type pour T_p. On utilise une loi normale pour retrouver la variabilité des mesures. On simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire suivant une loi normale centrée sur 0 et d'écart-type $u(T_p) = 0{,}1\,^{\circ}\mathrm{C}$.

  Avec Python, on peut simuler une variable normale en utilisant la syntaxe :

  ```
  Tp+rd.normal(0, u_Tp)
  ```

  On peut écrire également :

  ```
  rd.normal(Tp, u_Tp)
  ```

3. Pour estimer l'incertitude-type de h, on utilise la méthode de simulation de Monte-Carlo :

- On réalise N tirages aléatoires.
- Pour chaque tirage, on simule un processus aléatoire pour les grandeurs U, I, S, T_p, T_e. Pour cela, on ajoute aux valeurs de U, I, S, T_p, T_e des valeurs aléatoires dont l'écart-type est égal à l'incertitude-type. On calcule la grandeur h en utilisant les variables simulées précédemment et on stocke le résultat dans le tableau `tab_h`.
- On calcule la moyenne et l'écart-type des valeurs du tableau `tab_h`.

Remarque : Voir exercice 1.1 « Incertitudes - Évaluation de type A » pour le calcul de l'écart-type avec $N - 1$ au dénominateur.

4.

```
import numpy as np          #bibliothèque numpy renommée np
import numpy.random as rd   #module numpy.random renommé rd
#mesures expérimentales
U=44.1
I=0.09
```

```
L=5.0e-2
l=2.8e-2
Tp=36.3
Te=22
#calcul de la demi-étendue
delta_U=(0.5/100*U+0.02)
delta_I=(1/100*I+0.003)
delta_L=0.5/1000
delta_l=0.5/1000
delta_Te=0.5
u_Tp=0.1            #incertitude-type
S=l*L               #surface d'échange avec l'air
h=U*I/((Tp-Te)*S)

N=10000             #nombre de tirages
tab_h=np.zeros(N)   #tableau contenant N valeurs nulles
for i in range(N):  #i varie de 0 inclus à N exclu
    val_U=U+delta_U*(2*rd.random()-1)
    #2*rd.random()-1 est un nombre aléatoire compris
    #entre -1 et 1
    val_I=I+delta_I*(2*rd.random()-1)
    val_L=L+delta_L*(2*rd.random()-1)
    val_l=l+delta_l*(2*rd.random()-1)
    val_Te=Te+delta_Te*(2*rd.random()-1)
    val_Tp=rd.normal(Tp, u_Tp)
    tab_h[i]=val_U*val_I/((val_Tp-val_Te)*val_L*val_l)

hmes=np.mean(tab_h)             #moyenne du tableau tab_h
u_hmes=np.std(tab_h, ddof=1)    #écart-type du tableau tab_h
                                #avec N-1 au dénominateur
```

Cours : Opérations sur les tableaux numpy

On considère deux tableaux A (N valeurs aléatoires suivant une loi uniforme) et B (N valeurs aléatoires suivant une loi normale). On souhaite définir le tableau C tel que `C[i] = A[i] × B[i]` pour chaque indice i compris entre 0 et $N - 1$.

Deux possibilités :

1. Utilisation d'une boucle `for`

```
import numpy.random as rd  #module numpy.random renommé rd
import numpy as np         #bibliothèque numpy renommée np
N=10
borne_inf=3
borne_sup=5
valeur_centrale=4
écart_type=0.5
A=np.zeros(N)      #tableau comportant N valeurs nulles
B=np.zeros(N)      #tableau comportant N valeurs nulles
```

```
C=np.zeros(N)      #tableau comportant N valeurs nulles
for i in range(N): #i varie entre 0 inclus et N exclu
    A[i]=rd.uniform(borne_inf, borne_sup)
    #variable aléatoire suivant une loi uniforme
    #(ou rectangulaire) entre borne_inf et borne_sup
    B[i]=rd.normal(valeur_centrale, écart_type)
    #variable aléatoire suivant une loi normale (ou gaussienne)
    #centrée sur valeur_centrale et d'écart-type incertitude_type
    C[i]=A[i]*B[i]
```

2. Pas d'utilisation de boucle `for`

```
import numpy.random as rd  #module numpy.random renommé rd
N=10
borne_inf=3
borne_sup=5
A=rd.uniform(borne_inf, borne_sup, N)
#tableau de N variables aléatoires suivant une loi uniforme
#(ou rectangulaire) entre borne_inf et borne_sup
valeur_centrale=4
écart_type=0.5
B=rd.normal(valeur_centrale, incertitude_type, N)
#tableau de N variables aléatoires suivant une loi normale
#(ou gaussienne) centrée sur valeur_centrale
#et d'écart-type incertitude_type
C=A*B  #multiplication de deux tableaux élément par élément
       #attention : il ne s'agit pas de la multiplication
       #de matrices
```

Les symboles « + », « – », « * » et « / » permettent d'additionner, de soustraire, de multiplier ou de diviser un par un les éléments des tableaux qui doivent être de même dimension. Le temps de calcul est ainsi considérablement réduit par rapport à l'utilisation de boucles `for`. Le code Python est plus compact en écrivant directement :

```
C=A*B
```

5.

```
#chaque tableau contient N valeurs aléatoires suivant une
#loi uniforme
tab_U=rd.uniform(U-delta_U, U+delta_U, N)
tab_I=rd.uniform(I-delta_I, I+delta_I, N)
tab_L=rd.uniform(L-delta_L, L+delta_L, N)
tab_l=rd.uniform(l-delta_l, l+delta_l, N)
tab_Te=rd.uniform(Te-delta_Te, Te+delta_Te, N)

#le tableau tab_Tp contient N valeurs aléatoires suivant une
#loi normale
tab_Tp=rd.normal(Tp, u_Tp, N)

#tableau tab_h calculé à partir des autres tableaux
tab_h=tab_U*tab_I/((tab_Tp-tab_Te)*tab_l*tab_L)

hmes=np.mean(tab_h)              #moyenne du tableau tab_h
u_hmes=np.std(tab_h, ddof=1)     #écart-type du tableau tab_h
                                 #avec N-1 au dénominateur
```

6. On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat de la mesure de h :

- $h = 198{,}5$ W.m^{-2}.K^{-1} et $u(h) = 7{,}0$ W.m^{-2}.K^{-1}
- $h = 198{,}5 \pm 7{,}0$ W.m^{-2}.K^{-1} (ce qui suit $\pm$ représente l'incertitude-type)

Remarque :

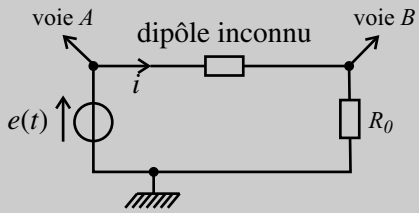
Pour calculer le résultat de la mesure de h, on a deux possibilités :

- Utilisation des grandeurs mesurées U, I, L, ℓ, T_e et T_P. On calcule `h=U*I/((Tp-Te)*S)` avant même de réaliser la simulation de Monte-Carlo. C'est l'approche classique.
- Utilisation des données simulées avec la méthode de Monte-Carlo. On calcule la moyenne des valeurs du tableau `tab_h`. C'est l'approche moderne, quasi bayésienne.

La différence entre les deux approches est en général très faible. On peut choisir indifféremment l'une ou l'autre approche.

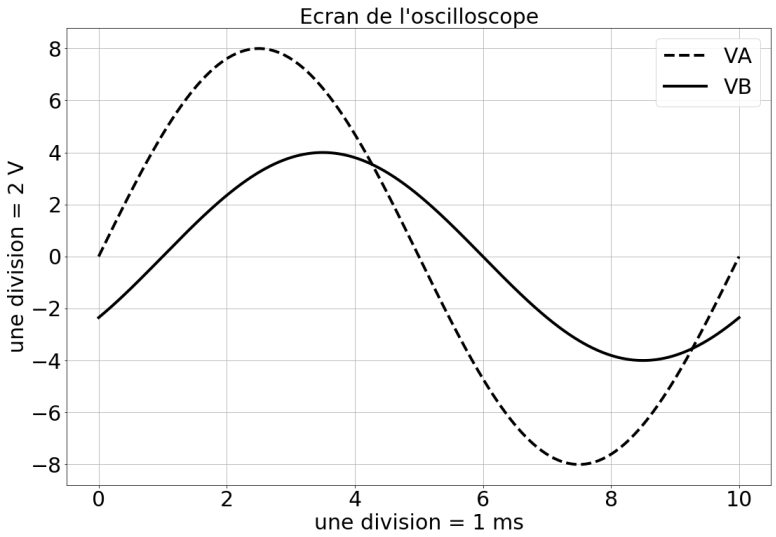
Exercice 1.3 : Variabilité - Mesure de *R* et *L*

On considère le montage suivant en régime sinusoïdal forcé. La résistance R_0 vaut 150 Ω avec une précision de 1 %. On observe sur un écran d'oscilloscope les deux tensions V_A et V_B. On peut considérer que les lectures sur l'oscilloscope se font avec une précision égale à 1/10e de division. On appelle V_{Am} et V_{Bm} les amplitudes des tensions V_A et V_B sur l'oscilloscope, t_1 le décalage temporel entre les deux tensions et T la période de la tension $e\,(t)$ délivrée par le GBF.




```
import numpy.random as rd  #module numpy.random renommé rd
rd.random() #renvoie un nombre aléatoire suivant une loi uniforme
   #dans la plage semi-ouverte [0.0, 1.0[
rd.uniform(borne_inf, borne_sup) #renvoie un nombre aléatoire
   #suivant une loi uniforme (ou rectangulaire)
   #entre borne_inf et borne_sup
rd.normal(valeur_centrale, incertitude_type #renvoie un nombre
   #aléatoire suivant une loi normale (ou gaussienne) centrée
   #sur valeur_centrale et d'écart-type incertitude_type
```

```
rd.uniform(borne_inf, borne_sup, N) #renvoie un tableau contenant
   #N variables aléatoires suivant une loi uniforme
   #(ou rectangulaire) entre borne_inf et borne_sup
rd.normal(valeur_centrale, incertitude_type, N) #renvoie un
   #tableau contenant N variables aléatoires suivant une loi
   #normale (ou gaussienne) centrée sur valeur_centrale
   #et d'écart-type incertitude_type
import numpy as np   #bibliothèque numpy renommée np
np.std(data, ddof=1) #renvoie l'écart-type du tableau data avec N-1
   #au dénominateur pour un tableau contenant N valeurs
```



1. On cherche à mettre l'impédance sous la forme $\underline{Z} = R + jL\omega$. Montrer que $R = R_0 \left(\dfrac{V_{Am}}{V_{Bm}} \cos\left(\dfrac{2\pi t_1}{T} \right) - 1 \right)$ et $L = \dfrac{TR_0}{2\pi} \dfrac{V_{Am}}{V_{Bm}} \sin\left(\dfrac{2\pi t_1}{T} \right)$.

2. Calculer les incertitudes-types des grandeurs V_{Am} et R_0. Écrire, avec un nombre adapté de chiffres significatifs, les résultats des mesures de R_0 et V_{Am}.

3. On cherche à mesurer R et L avec les relations précédentes. Utilise-t-on une loi uniforme ou une loi normale pour simuler un processus aléatoire pour les grandeurs R_0, T, t_1, V_{Bm} et V_{Am} ?

4. Expliquer la méthode de simulation de Monte-Carlo pour calculer les incertitudes-types de R_0 et L.

5. Écrire un programme Python permettant de calculer les valeurs moyennes et les incertitudes-types de R_0 et L en utilisant des boucles `for`.

6. Écrire un programme Python permettant de calculer les valeurs moyennes et les incertitudes-types de R_0 et L sans utiliser de boucle `for`.

7. Lors de l'exécution du programme Python, on obtient $R = 92{,}61799$; $u(R) = 10{,}093$; $L = 0{,}28055$ et $u(L) = 0{,}01682$. Écrire, avec un nombre adapté de chiffres significatifs, les résultats des mesures de R et L.

Analyse du problème

La méthode de Monte-Carlo est très pratique à utiliser pour la propagation des incertitudes. On peut ainsi déterminer l'incertitude-type d'une grandeur calculée à partir des grandeurs dont on connaît les incertitudes-types.

1. La période du signal vaut T. La voie B est en retard par rapport à la voie A. On choisit la tension V_A comme origine des phases. On appelle t_1 le décalage temporel entre les deux signaux. Le retard de phase de la voie B par rapport à la voie A vaut $\varphi = 2\pi\dfrac{t_1}{T} > 0$. On pose : $V_A(t) = V_{Am}\cos(\omega t)$, $\underline{V_A} = V_{Am}$, $V_B(t) = V_{Bm}\cos(\omega t - \varphi)$ et $\underline{V_B} = V_{Bm}\exp(-j\varphi)$. On appelle $\underline{Z}$ l'impédance du dipôle inconnu.

On peut appliquer un diviseur de tension : $\underline{V_B} = \underline{V_A}\dfrac{R_0}{R_0 + \underline{Z}}$, d'où $\underline{V_B}\left(R_0 + \underline{Z}\right) = \underline{V_A}R_0$, soit $R_0 + \underline{Z} = \dfrac{\underline{V_A}}{\underline{V_B}}R_0$.

On a alors : $\underline{Z} = R_0\left(\dfrac{\underline{V_A}}{\underline{V_B}} - 1\right)$. Comme $\dfrac{\underline{V_B}}{\underline{V_A}} = \dfrac{V_{Bm}}{V_{Am}}\exp(-j\varphi)$, on en déduit que :

$$\underline{Z} = R_0\left(\frac{V_{Am}}{V_{Bm}}\exp(j\varphi) - 1\right) = R_0\left(\frac{V_{Am}}{V_{Bm}}\cos(\varphi) - 1\right) + jR_0\frac{V_{Am}}{V_{Bm}}\sin(\varphi)$$

On peut alors identifier à $\underline{Z} = R + jL\omega$ avec $\omega = \dfrac{2\pi}{T}$. Finalement, on obtient :

$$R = R_0\left(\frac{V_{Am}}{V_{Bm}}\cos\left(2\pi\frac{t_1}{T}\right) - 1\right)$$

$$L = \frac{TR_0}{2\pi}\frac{V_{Am}}{V_{Bm}}\sin\left(2\pi\frac{t_1}{T}\right)$$

Il faudra bien vérifier que les deux grandeurs calculées sont positives.

Cours :

On se place dans le cas où on n'a réalisé qu'une observation unique. Afin de retrouver la variabilité intrinsèque à la mesure, on définit un intervalle dans lequel on est raisonnablement certain que le résultat de la mesure se trouve.

Deux possibilités pour retrouver la variabilité de la mesure :

- Si on ne connaît que la limité basse et la limite haute pour la grandeur considérée (par exemple $T, t_1, R_0, V_{Am}, V_{Bm}$), alors on suppose que la **distribution est uniforme** entre ces deux bornes. On utilise une loi uniforme que l'on appelle également rectangulaire.
- Si on connaît l'incertitude-type associée à cette mesure, alors on suppose que la **distribution est gaussienne**. On utilise une loi normale dont l'écart-type est l'incertitude-type.

2. La précision vaut $\Delta(R_0) = \dfrac{1}{100} \times 150 = 1{,}5\ \Omega$. On peut raisonnablement penser que la résistance « réelle » est comprise entre $150{,}0 - 1{,}5\ \Omega$ et $150{,}0 + 1{,}5\ \Omega$. La demi-étendue de cet intervalle vaut $\Delta(R_0)$. Comme on n'a aucune autre information que la limite basse et la limite haute pour la résistance, on suppose que la répartition est uniforme entre ces deux bornes. On utilise une loi uniforme ou rectangulaire. L'incertitude-type de la résistance vaut $u(R_0) = \dfrac{\Delta(R_0)}{\sqrt{3}} = 0{,}87\ \Omega$.

On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat de la mesure de V_{Am} :

- $R_0 = 150{,}00\ \Omega$ et $u(R_0) = 0{,}87\ \Omega$
- $R_0 = 150{,}00 \pm 0{,}87\ \Omega$ (ce qui suit $\pm$ représente l'incertitude-type)

Remarque : On ne connaît pas la valeur vraie de la résistance R_0.

On mesure sur l'écran de l'oscilloscope 4 divisions (soit 8 V) avec une précision égale à 1/10e de carreau, soit 0,2 V. On peut raisonnablement penser que V_{Am} est compris entre $8{,}0 - 0{,}2$ V et $8{,}0 + 0{,}2$ V. La demi-étendue de cet intervalle vaut $\Delta(V_{Am})$. Comme on n'a aucune autre information que la limite basse et la limite haute pour V_{Am}, on suppose que la répartition est uniforme entre ces deux bornes. On utilise une loi uniforme ou rectangulaire. L'incertitude-type de V_{Am} vaut $u(V_{Am}) = \frac{\Delta(V_{Am})}{\sqrt{3}} = 0{,}12$ V.

On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat de la mesure de V_{Am} :

- $V_{Am} = 8{,}00$ V et $u(V_{Am}) = 0{,}12$ V
- $V_{Am} = 8{,}00 \pm 0{,}12$ V (ce qui suit $\pm$ représente l'incertitude-type)

Remarques : On a deux zéros derrière la virgule. Dans la valeur mesurée, on perd la notion de chiffres significatifs. On ajoute deux zéros par commodité de lecture. Le deuxième zéro n'a plus de rapport avec la résolution de l'instrument. Il ne faut donc pas lui attribuer de sens physique. L'instrument de mesure ne fixe pas le nombre de chiffres significatifs de la valeur mesurée.

Autres mesures effectuées sur l'écran de l'oscilloscope :

- $T = 10$ ms au $1/10$e de carreau, soit une précision de $0{,}1$ ms
- $t_1 = 1$ ms au $1/10$e de carreau, soit une précision de $0{,}1$ ms
- $V_{Bm} = 4$ V au $1/10$e de carreau, soit une précision de $0{,}2$ V

3. On ne connaît que les limites basses et hautes pour les grandeurs $T, t_1, R_0, V_{Am}, V_{Bm}$. On utilise une loi uniforme pour retrouver la variabilité des mesures : on simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire ayant les caractéristiques de l'incertitude-type évaluée.

Par exemple pour V_{Am}, la valeur mesurée vaut 8 V. On simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire comprise entre $-\Delta(V_{Am})$ et $\Delta(V_{Am})$ suivant une loi uniforme. Avec Python, on peut simuler une variable aléatoire suivant une loi uniforme en utilisant la syntaxe :

```
VAm + rd.uniform(-delta_VAm, delta_VAm)
```

On peut écrire également :

```
VAm+delta_VAm*(2*rd.random()-1)
```

4. Pour estimer les incertitudes-types de R et L, on utilise la méthode de simulation de Monte-Carlo :

- On réalise N tirages aléatoires.
- Pour chaque tirage, on simule un processus aléatoire pour les grandeurs $T, t_1, R_0, V_{Am}, V_{Bm}$. Pour cela, on ajoute aux valeurs de $T, t_1, R_0, V_{Am}, V_{Bm}$ des valeurs aléatoires dont l'écart-type est égal à l'incertitude-type. On calcule les grandeurs R et L en utilisant les variables simulées précédemment et on stocke le résultat dans les tableaux `tab_R` et `tab_L`.
- On calcule les moyennes et les écarts-types des valeurs des tableaux `tab_R` et `tab_L`.

Remarque : Voir exercice 1.1 « Incertitudes - Évaluation de type A » pour le calcul de l'écart-type avec $N - 1$ au dénominateur

5.

```
import numpy as np          #bibliothèque numpy renommée np
import numpy.random as rd   #module numpy.random renommé rd
#mesures expérimentales
T=10e-3
t1=1e-3
VAm=8
VBm=4
R0=150
```

```
#calcul de la demi-étendue
delta_T=1/10*1e-3    #précision 1/10e de carreau
delta_t1=1/10*1e-3   #précision 1/10e de carreau
delta_R0=1/100*R0    #précision 1%
delta_VAm=1/10*2     #précision 1/10e de carreau
delta_VBm=1/10*2     #précision 1/10e de carreau

R=R0*(VAm/VBm*np.cos(2*np.pi*t1/T)-1)
L=R0*VAm/VBm*np.sin(2*np.pi*t1/T)/(2*np.pi/T)

N=10000              #nombre de tirages
tab_R=np.zeros(N)    #tableau contenant N valeurs nulles
tab_L=np.zeros(N)    #tableau contenant N valeurs nulles
for i in range(N):   #i varie de 0 inclus à N exclu
    val_T=T+delta_T*(2*rd.random()-1)
    #2*rd.random()-1 est un nombre aléatoire compris
    #entre -1 et 1
    val_t1=t1+delta_t1*(2*rd.random()-1)
    val_R0=R0+delta_R0*(2*rd.random()-1)
    val_VAm=VAm+delta_VAm*(2*rd.random()-1)
    val_VBm=VBm+delta_VBm*(2*rd.random()-1)
    tab_R[i]=val_R0*(val_VAm/val_VBm*np.cos(2*np.pi*val_t1\
       /val_T)-1)
    tab_L[i]=val_R0*val_VAm/val_VBm*np.sin(2*np.pi*val_t1\
       /val_T)/(2*np.pi/val_T)

#Moyenne et écart-type de Rmes et Lmes
Rmes=np.mean(tab_R)              #moyenne du tableau tab_R
u_Rmes=np.std(tab_R, ddof=1)     #écart-type du tableau tab_R
                                 #avec N-1 au dénominateur
Lmes=np.mean(tab_L)              #moyenne du tableau tab_L
u_Lmes=np.std(tab_L, ddof=1)     #écart-type du tableau tab_L
                                 #avec N-1 au dénominateur
```

Cours :

Voir exercice 1.2 « Variabilité d'une grandeur composée » pour les opérations réalisées avec des tableaux Python.

6.

```
#chaque tableau contient N valeurs aléatoires suivant une
#loi uniforme
tab_T=rd.uniform(T-delta_T, T+delta_T, N)
tab_t1=rd.uniform(t1-delta_t1, t1+delta_t1, N)
tab_R0=rd.uniform(R0-delta_R0, R0+delta_R0, N)
tab_VAm=rd.uniform(VAm-delta_VAm, VAm+delta_VAm, N)
tab_VBm=rd.uniform(VBm-delta_VBm, VBm+delta_VBm, N)
```

```
#tableaux tab_R et tab_L calculés à partir des autres
#tableaux
tab_R=tab_R0*(tab_VAm/tab_VBm*np.cos(2*np.pi*tab_t1/tab_T)-1)
tab_L=tab_R0*tab_VAm/tab_VBm*np.sin(2*np.pi*tab_t1/tab_T) \
  /(2*np.pi/tab_T)

Rmes=np.mean(tab_R)              #moyenne du tableau tab_R
u_Rmes=np.std(tab_R, ddof=1)     #écart-type du tableau tab_R
                                 #avec N-1 au dénominateur
Lmes=np.mean(tab_L)              #moyenne du tableau tab_L
u_Lmes=np.std(tab_L, ddof=1)     #écart-type du tableau tab_L
                                 #avec N-1 au dénominateur
```

7. On utilise deux chiffres significatifs pour l'incertitude-type. Plusieurs écritures possibles du résultat des mesures :

- $R = 93\ \Omega$ et $u(R) = 10\ \Omega$
- $R = 93 \pm 10\ \Omega$ (ce qui suit $\pm$ représente l'incertitude-type)
- $L = 0{,}281$ H et $u(R) = 0{,}017$ H
- $L = 0{,}281 \pm 0{,}017$ H (ce qui suit $\pm$ représente l'incertitude-type)

Remarque : Pour calculer le résultat des mesures de R et L, on a deux possibilités :

- Utilisation des grandeurs mesurées $T, t_1, R_0, V_{Am}, V_{Bm}$. On calcule R et L avant même de réaliser la simulation de Monte-Carlo. C'est l'approche classique.
- Utilisation des données simulées avec la méthode de Monte-Carlo. On calcule la moyenne des valeurs des tableaux `tab_R` et `tab_L`. C'est l'approche moderne, quasi bayésienne.

La différence entre les deux approches est en général très faible. On peut choisir indifféremment l'une ou l'autre approche.

Exercice 1.4 : Régression linéaire - Incertitudes sur les paramètres du modèle

On considère 6 mesures expérimentales de x et y. Les valeurs de x_i sont connues avec certitude. Pour chaque valeur de y_i, on connaît l'incertitude-type $u(y_i)$.

x	6	7	8	9	10	11
y	1,9	6,1	7,8	8,5	12,1	15,3
$u(y)$	5	1,3	1,1	1,2	1,1	1,5

On cherche à modéliser les données expérimentales par la droite affine : $y = ax + b$.

```
import numpy.random as rd  #module numpy.random renommé rd
import numpy as np         #bibliothèque numpy renommée np
rd.uniform(borne_inf, borne_sup, N) #renvoie un tableau contenant
  #N variables aléatoires suivant une loi uniforme
  #(ou rectangulaire) entre borne_inf et borne_sup
rd.normal(valeur_centrale, incertitude_type, N) #renvoie un
  #tableau contenant N variables aléatoires suivant une loi
  #normale (ou gaussienne) centrée sur valeur_centrale
  #et d'écart-type incertitude_type
np.std(data, ddof=1) #renvoie l'écart-type du tableau data avec N-1
  #au dénominateur pour un tableau contenant N valeurs
np.polyfit(x,y,1)  #régression linéaire de y en fonction de
  #x : y = a x + b. La fonction renvoie [a, b]
```

Les fonctions suivantes permettent le tracé de fonctions :

```
import matplotlib.pyplot as plt  #module matplotlib.pyplot
                                 #renommé plt
plt.figure()                     #nouvelle fenêtre graphique
plt.plot(x, y, color='r', linewidth =3, marker='o', label='points')
    #color : choix de la couleur
    #('r' : red, 'g' : green, 'b' : blue, 'black' : black)
    #linewidth : épaisseur du trait
    #marker : différents symboles '+', '.', 'o', 'v'
    #linestyle : style de la ligne : '-' (ligne continue),
    #'-' (ligne discontinue),
    #':' (ligne pointillée)
plt.plot(x, y, '*') #points non reliés représentés par '*'
plt.errorbar(x, y, xerr=u_x, yerr=u_y , fmt='*')  #affiche les
    #barres d'incertitude pour x et y avec u_x et u_y
    #les incertitudes-types pour x et y. Points non reliés
    #représentés par '*'
plt.grid()          #affiche la grille
plt.title('Titre')  #ajout d'un titre
plt.xlabel('axe x') #affiche 'axe x' en abscisse d'un graphique
plt.ylabel('axe y') #affiche 'axe y' en ordonnée d'un graphique
plt.axis ([xmin, xmax, ymin, ymax])  #précise les bornes pour les
                    #abscisses et les ordonnées
plt.legend(['courbe 1', 'courbe 2']) #permet de légender
                                     #les courbes
plt.show()          #affiche la figure à l'écran
```

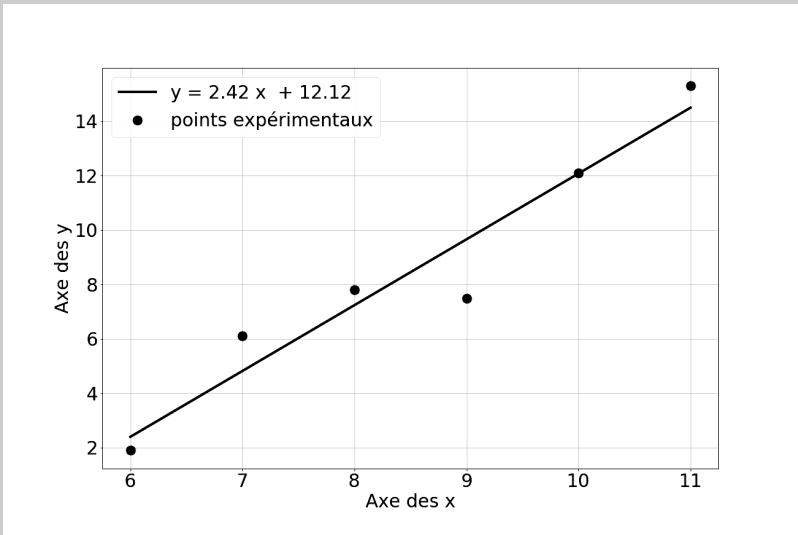
1. Écrire un programme Python permettant de :

- calculer a et b ;
- représenter graphiquement les points expérimentaux et la droite de régression.

Le graphique doit avoir les caractéristiques suivantes :

- courbe représentative de la droite de régression : couleur rouge. Légende : '$y = ax + b$' ;
- courbe représentative des points expérimentaux. Points non reliés représentés par 'o', couleur bleue. Légende : 'points expérimentaux' ;
- afficher la grille ;
- afficher 'Axe des x' pour l'axe des abscisses et 'Axe des y' pour l'axe des ordonnées.

2. Est-ce que la droite $y = 2.42x + 12.12$ est un modèle acceptable ?

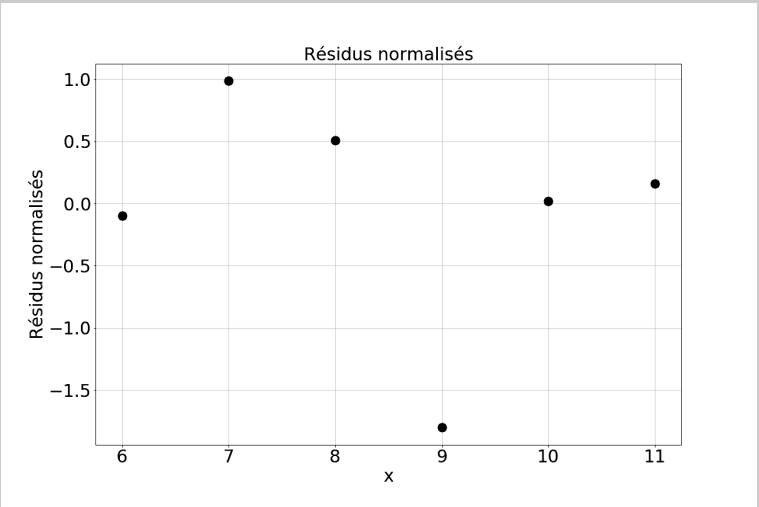


3. Le résidu d'un point d'indice i est défini par $y_i - (ax_i + b)$. Le résidu normalisé (ou écart normalisé) d'un point d'indice i est défini par $\dfrac{y_i - (ax_i + b)}{u(y_i)}$. Écrire un programme Python permettant de :

- Calculer la liste des résidus $y_i - (ax_i + b)$;
- Représenter graphiquement la liste des résidus normalisés $\dfrac{y_i - (ax_i + b)}{u(y_i)}$ en fonction de x_i.
  Le graphique doit avoir les caractéristiques suivantes :
  – points non reliés représentés par 'o', couleur bleue ;
  – titre : 'Résidus normalisés' ;
  – afficher 'x' pour l'axe des abscisses et 'Résidus normalisés' pour l'axe des ordonnées.
- Représenter graphiquement la droite de régression avec les barres d'incertitude $\pm u$. Le graphique doit avoir les caractéristiques suivantes :

– courbe représentative de la droite de régression : couleur rouge. Légende : '$y = ax + b$' ;
– courbe représentative des points expérimentaux avec les barres d'incertitude. Points non reliés représentés par 'o', couleur bleue. Légende = '(barres d'erreur = inc. type)' ;
– afficher la grille ;
– afficher 'x' pour l'axe des abscisses et 'y' pour l'axe des ordonnées.

4. Est-ce que la droite $y = 2.42x + 12.12$ est un modèle acceptable ?



5. Pour estimer les incertitudes pour a et b, on utilise la simulation de Monte-Carlo qui consiste à simuler un processus aléatoire de variation des valeurs de y.
On procède comme dans l'exercice 1.2 « Variabilité d'une grandeur composée » : on simule de nouvelles données expérimentales sur la base des valeurs effectivement mesurées en y ajoutant une valeur aléatoire ayant les caractéristiques de l'incertitude-type évaluée. À chaque simulation, on calcule la pente et l'ordonnée à l'origine. On obtient une collection de pentes et d'ordonnées à l'origine. L'écart-type expérimental des pentes est une estimation de leur incertitude-type, de même pour les ordonnées à l'origine.
Écrire un programme Python permettant d'estimer les incertitudes-types de a et b.

Analyse du problème

La fonction `np.polyfit` permet de calculer le coefficient directeur et l'ordonnée à l'origine de la droite de régression qui passe au plus près des points expérimentaux. L'analyse graphique intégrant les barres d'incertitude ou l'analyse des écarts normalisés permet de valider ou non le modèle linéaire.

1.

```
import numpy as np                  #bibliothèque numpy
                                    #renommée np
import matplotlib.pyplot as plt     #module matplotlib.pyplot
                                    #renommé plt
x = np.array([6, 7, 8, 9, 10, 11])
y = np.array([1.9, 6.1, 7.8, 7.5, 12.1, 15.3])
a, b = np.polyfit(x, y, 1) #régression linéaire de y en
                           #fonction de x: y=ax+b
xfit = np.linspace(np.min(x), np.max(x),2) #tableau de
                                           #deux points
yfit=a*xfit+b           #tableau de deux points
plt.figure()            #nouvelle fenêtre graphique
plt.plot(xfit, yfit, color='red')
plt.grid()              #affiche la grille
plt.xlabel('Axe des x') #affiche 'x' en abscisses du graphique
plt.ylabel('Axe des y') #affiche 'y' en ordonnées du graphique
plt.plot(x, y, 'o', color='blue')
                        #points non reliés représentés par 'o'
                        #couleur bleue
plt.legend(['y=ax+b', 'points expérimentaux'])
plt.show()              #affiche la figure à l'écran
```

2.

Comme on n'a pas déterminé les barres d'incertitude pour chaque point, on ne peut pas affirmer que la droite de régression décrit correctement les données expérimentales. Pour valider le modèle affine, on peut tracer les résidus normalisés ou représenter la droite de régression avec les barres d'incertitude $\pm u$.

3.

```
u_y = np.array([5, 1.3, 1.1, 1.2, 1.1, 5]) #tableau des
                                           #incertitudes-types
Residus=y-(a*x+b) #tableau des résidus
plt.figure()      #nouvelle fenêtre graphique
plt.xlabel('Axe des x')
plt.ylabel('Axe des y')
plt.plot(xfit, yfit, color='r')
plt.errorbar(x, y, yerr=u_y, fmt='o', color='blue')
plt.legend(['y=ax+b', "(barres d'erreur = inc. type)"])
plt.grid()       #affiche la grille
plt.show()       #affiche la figure à l'écran
z=Residus/u_y    #tableau des écarts normalisés
plt.figure()     #nouvelle fenêtre graphique
plt.title('Résidus normalisés')
plt.plot(x, z, 'o', color='Blue')
plt.grid()       #affichage de la grille
plt.show()       #affiche la figure à l'écran
```