## Contenu :



## E3A MPI 2023 – Peser la Terre

*Cette partie fait appel à de la programmation en langage Python, dont un certain nombre de " fonctions " sont rappelées en fin d'énoncé pages 17 à 19.*

### III. 1 - Principe

On dit que l'interaction gravitationnelle permet de peser les astres.

**Q22.** Justifier que l'action " peser un astre " est l'action de mesure de sa masse et non de son poids.

L'observation de satellites terrestres artificiels permet de déterminer la masse terrestre $M_T$.

On rappelle que ces satellites ont des masses très inférieures à celle de la Terre et des trajectoires *a priori* elliptiques dont un foyer est le centre d'inertie de la Terre $O_T$. On utilise pour cela la relation, établie par Kepler en 1619 dans l'ouvrage *Harmonices Mundi*, connue sous le nom de troisième loi de Kepler : $\frac{a^3}{T^2} = \frac{G M_T}{4\pi^2}$. Dans cette expression, a est le demi-grand axe de l'ellipse trajectoire et T la période de révolution du satellite.

**Q23.** Faire un schéma clair de l'ellipse, trajectoire du satellite, en y positionnant le foyer $O_T$ et le demi-grand axe a. Faire figurer également les points particuliers P et A de la trajectoire, appelés respectivement «périgée» et «apogée», ainsi que les distances $r_P$ et $r_A$ de ces points au foyer $O_T$. Exprimer le demi-grand axe a en fonction de $r_P$ et $r_A$.

On veut obtenir la valeur de $M_T$ grâce à une régression linéaire à partir des couples de valeurs (a, T) obtenues pour 9 satellites différents. Pour cela, on trace la courbe Y = F(X). On s'attend à une droite dont la pente, notée alpha, permet de calculer $M_T$.

**Q24.** Exprimer Y en fonction de a et X en fonction de T. Exprimer littéralement $M_T$ en fonction de alpha et des valeurs constantes apparaissant dans la troisième loi de Kepler.

Par la suite, on attend un résultat en kg pour la valeur numérique de $M_T$.

### III. 2 - Étude de données orbitales

On trouve sur le web plusieurs sites (par exemple le site *Live realtime satellite tracking and predictions*) qui donnent en temps réel les positions de centaines de satellites dans le référentiel terrestre.

On suit 9 satellites. Pour un satellite, on dispose alors de données telles que sa période (en minutes), et, en temps réel pour $N_d$ dates, son altitude (en km), ainsi que sa latitude et sa longitude, en coordonnées terrestres. Après extraction des données, on obtient, pour le groupe des 9 satellites, une liste DATA composées de 9 listes. Chacune des 9 listes contient, pour chaque satellite, les valeurs suivantes : numéro, nom, T, t, lat, long, alt.

*Description du groupe de listes pour un satellite donné* :

| *Liste :* | *description :* | *format :* |
|---|---|---|
| numero | numéro du satellite étudié | entier de 1 à 9 |
| nom | nom du satellite | chaîne de caractères |
| T | période, en minute | flottant positif |
| t | liste des dates de mesures des positions, | liste de $N_d$ flottants positifs |
| lat | liste des latitudes, en degré, | liste de $N_d$ flottants |
| long | liste des longitudes, en degré, | liste de $N_d$ flottants |
| alt | liste des altitudes, en kilomètre, | liste de $N_d$ flottants positifs |

On considère les instructions suivantes (document 1) :

```
n_sat=[ ]
for i in range(9) :
        n_sat.append(DATA[i][0])
```

Document 1

**Q25.** Expliquer ce que fait ce morceau de code.

**Q26.** Écrire les instructions pour extraire de DATA la liste T_sat contenant les 9 périodes exprimées en minutes.

**Q27.** Écrire la fonction **demiGrandAxe**(DATA) qui permet d'obtenir la liste a_sat contenant les demi-grands axes a des 9 trajectoires exprimées en km.

On obtient alors les listes décrites dans le **document 2** :

```
n_sat =   [ 1      , 2      , 3      , 4      , 5      , 6      , 7      , 8      , 9 ]
          # Numéro du satellite

T_sat =   [ 92.9   ,1436.1  ,101.9   ,1436.2  ,631.1   ,1214.4  ,93.     ,118.4   ,104.2 ]
          # Période de révolution T en minutes

a_sat =   [ 794.   ,42164.  ,7228.   ,42167.  ,24371.  ,37704.  ,6825.   ,7988.   ,7333.]
          # Demi-grand axe a en km
```

Document 2

**Q28.** Écrire la fonction **XY** dont les arguments sont les listes T_sat et a_sat, qui retourne les tableaux de type numpy.ndarray, X et Y dans le tuple (X,Y).

On calcule les paramètres alpha et beta de la droite de régression linéaire obtenues à partir des valeurs X et Y, afin d'obtenir le tableau Y_mod tel que : Y_mod = alpha * X + beta.

**Q29.** Écrire les instructions pour obtenir le tableau [alpha, beta] des paramètres de la régression linéaire. Quelle valeur attend-on pour l'ordonnée à l'origine beta ? Justifier.

Grâce à la bibliothèque **matplotlib.pyplot**, le code du document 3 permet de tracer Y = F(X) pour les 9 satellites enregistrés ainsi qu'une droite de régression linéaire.

```
plt.figure()
plt.plot( A , B ," bo ", label= " Données ")
plt.plot( C , D ," r- ", label= " Régression linéaire ")
plt.title(" Exploitation de la troisième loi de Képler ")
plt.xlabel( " E " )
plt.ylabel( " F " )
plt.legend()
plt.grid(True)
plt.show()
```

Document 3

**Q30.** Expliciter les termes **A**, **B**, **C**, **D**, **E**, et **F**.

Un élève obtient la courbe de la figure 12 et s'en satisfait.

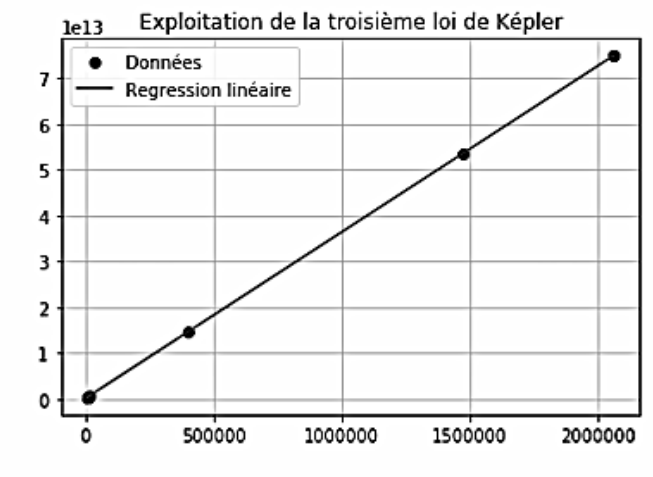


Figure 12 : graphique d'un élève

Hélas, ce graphique n'est pas validé par le professeur, pour plusieurs raisons.

**Q31.** Quelle(s) précaution(s) l'élève a-t-il oubliée(s) de prendre en écrivant la fonction **XY**, puis en écrivant le code décrit sur le document 3 ?

**Q32.** Expliquer pourquoi seuls 4 points semblent apparaître sur le graphique, alors que les 9 valeurs ont bien été entrées dans le code.

### III. 3 - Précision du résultat

Cette méthode permet une première valeur pour $M_T$.

Les données précédemment utilisées sont connues avec une " marge " e_val, c'est-à-dire qu'une valeur associée à la mesure val se situe dans l'intervalle [val – e_val , val + e_val]. Il est donc légitime de s'intéresser à la précision du résultat obtenu pour $M_T$.

Afin de comparer la compatibilité de la valeur numérique obtenue avec la masse de la Terre publiée, on évalue l'incertitude-type u_M grâce à une simulation de Monte-Carlo. Pour cela, pour les 9 satellites retenus, on simule N_sim mesures de a et T. Les résultats de ces mesures sont répartis uniformément dans les intervalles [a – e_a , a + e_a] et [T – e_T , T + e_T]. On estime e_a = 0.01 % × a et e_T = 0.01 % × T.

Cette simulation procède à N_sim tirs aléatoires générés par une fonction Python de la bibliothèque **numpy.random**. Chaque tir aléatoire donne deux valeurs T_tir et a_tir, comprises dans les intervalles définis par la précision des mesures.

Pour chaque couple de valeurs tirées, on calcule les valeurs Y_tir et X_tir associées, puis les paramètres de la régression linéaire : la valeur de la pente, alpha_tir, et l'ordonnée à l'origine, beta_tir.

Finalement, les N_sim valeurs obtenues pour alpha_tir et beta_tir, permettent d'accéder, grâce à leurs moyennes et leurs écart-types, à M_T, u_M, beta_sim, et l'incertitude-type associée à beta, u_beta.

**Q33.** Proposer un ordre de grandeur pour N_sim.

La séquence d'instructions permettant d'obtenir les listes alpha_tir, beta_tir, puis M_tir, peut s'écrire comme décrit ci-dessous (document 4) :

```
# Initialisation ds listes
list_MT, list_beta, list_alpha = G, H, I
list_Y, list_X = J, K

# Tirages aléatoires
for i in range(L) :
        a_tir= M + (e_a) * rd.uniform(-1, 1, 9)        # Valeur aléatoire de a
        T_tir= N + (e_T) * rd.uniform(-1, 1, 9)        # Valeur aléatoire de T
        Y_tir= O
        X_tir= P

        list_Y.append(Q)          # AA
        list_X.append(R)

        p=np.polyfit(S,U,1)        # Régression linéaire
        list_alpha.append(V)
        list_beta.append(W)
        list_MT.append(Z)
```

Document 4

**Q34.** Expliciter les termes **G**, **H**, **I**, **J**, **K**, **L**, **M**, **N**, **O**, **P**, **Q**, **R**, **S**, **U**, **V**, **W**, **X**, **Y**, et **Z**. Proposer également le commentaire **AA**.

**Q35.** Écrire alors les instructions pour obtenir la masse M_T, l'incertitude-type associée u_M, ainsi que beta_sim et l'incertitude-type associée u_beta.

Le document 5 reproduit les résultats tels qu'ils sont affichés par le programme.

```
M_T = 5.975818959537378e + 24 kg
u_M = 7.708549319930416e + 20
beta_sim = −3.388893817989048e + 17
u_beta = 7.923053822786523e + 17
```

Document 5

On compare le résultat obtenu, M_T, avec la valeur publiée $M_T$. La littérature mentionne : $M_T$ = 5.97223 e+24 kg, $u_T$ = 5.9 e+20 kg.

**Q36.** Définir un critère de compatibilité. Les résultats obtenus (valeur et précision) à partir des données de DATA sont-ils compatibles avec la valeur publiée ?

## Données numériques

### Données relatives à la Terre

| | | |
|---|---|---|
| Rayon | $R_T =$ | $6,37 \cdot 10^3$ km |
| Masse totale | $M_T =$ | $5,97 \cdot 10^{24}$ kg |

### Constantes universelles

Constante de la gravitation universelle : $G = 6,67 \cdot 10^{-11}$ $m^3 \cdot kg^{-1} \cdot s^{-2}$

# Annexe : instructions en python

## Description des bibliothèques

### Numpy

Numpy est une bibliothèque pour langage de programmation Python destinée à manipuler des tableaux multidimensionnels de type numpy.ndarray ainsi que des fonctions mathématiques opérant sur ces tableaux.

On l'importe grâce à l'instruction : import numpy as np.

Quelques fonctions de la bibliothèque numpy :

**Tableau=np.array(Liste)**

*# transforme la liste Liste en tableau Tableau.*

**np.mean(Liste, axis=0)**

*# effectue la moyenne d'une liste de nombres ou d'un tableau.*

**np.std(Liste, axis=0, ddof=1)**

*# calcule l'estimateur de l'écart-type (déviation standard) d'une liste de nombres ou d'un tableau.*

**p=np.polyfit(x,y,deg)**

*# ajuste le nuage de points des valeurs de x et de y par un polynôme de degrés deg, et renvoie un tableau contenant les coefficients du polynôme.*

*Ainsi p(x) = p[0] * x**deg + ... + p[deg]* avec :

*p[0] : coefficient du terme de plus haut degré,*

*p[deg] : coefficient du terme constant .*

### Numpy.random

Numpy.random est un module de la bibliothèque numpy implémentant des générateurs de nombres aléatoires pour différentes distributions.

On l'importe grâce à l'instruction : import numpy.random as rd.

**rd.uniform(x, y, n)**

*# restitue un tableau de n nombres aléatoires uniformément répartis dans l'intervalle ]x,y[.*

### Matplotlib.pyplot

Matplotlib.pyplot est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.

On l'importe grâce à l'instruction : import matplotlib.pyplot as plt.

Quelques fonctions de la bibliothèque matplotlib.pyplot :

**plt.figure()**

*# crée une nouvelle figure.*

**plt.plot( x , y , " style ", label=" Légende ")**

*# trace x en abscisse et y en ordonnée.*

*Ce peut être des points isolés ou des segments reliés, selon le style choisi.*

**plt.plot( x , y , " bo ", label=" Points ")**

**plt.plot( x ,y , " r- ", label=" Droite ")**

**plt.title(" Titre ")**

*# indique Titre en titre du graphique.*

**plt.ylabel(" Ordonnée ")**

*# affiche Ordonnée sur l'échelle des ordonnées.*

**plt.xlabel(" Abscisse ")**

*# affiche Abscisse sur l'échelle des abscisses.*

**plt.grid(True)**

*# dessine un quadrillage.*

**plt.legend()**

*# affiche la légende.*

**plt.show()**

*# affiche la figure ;*

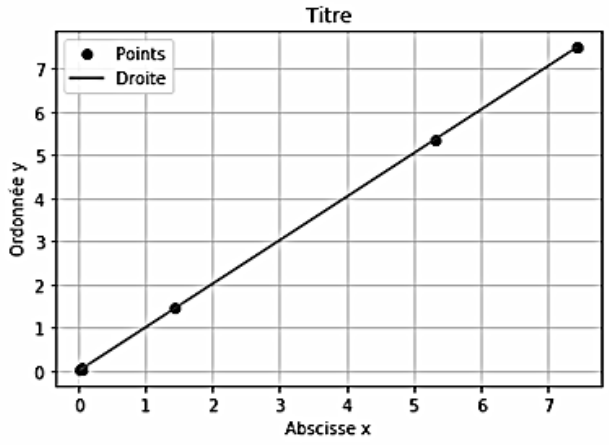


Figure 16 : *exemple de rendu de graphique*

## Opérations sur les listes

**liste_vide = [ ]**

*# création d'une liste vide.*

**liste.append(Element)**

*# rajoute Element en dernière position de la liste.*

**max(Liste_nombres)** *# renvoie le maximum d'une liste de nombres.*

**min(Liste_nombres)** *# renvoie le minimum d'une liste de nombres.*

**Sélection d'éléments d'une liste :**

Exemple :

Soit Liste = [ [5,[25,6,9] ], [ 8, [32,50,89] ] ]

Alors : **Liste[0][1] = [25,6,9]** et **Liste[1][1][2] = 89**

## Fonction

```
def function(arguments1, arguments2, . . . )
        ..... # instructions
        return Resultat
```

## Calculs

**np.pi** *# représentation par un flottant du nombre $\pi$ : np.pi = 3.141592653589793*

**Opérations mathématiques**

Exemples

*# multiplication* : 2*3 = 6 ; 2.*3 = 6.0

*# division* : 6/2 = 3 ; 6./3 = 2.0

*# exposant* : 10**3 = 1000 ; 2.**3 = 8.0

*# notation scientifique* : 2e+5 = 200000.0

# CCS1 TSI 2024 – Bouche d'un robot clarinettiste

Un ensemble de données et un formulaire utiles à la résolution du problème figurent en fin d'énoncé.

En novembre 2007, le laboratoire d'acoustique musicale de l'Université de Nouvelle-Galles du Sud (University of New South Wales, UNSW) situé à Sydney, s'est associé avec NICTA, un institut de recherche en télécommunication, pour construire un robot clarinettiste.

La fabrication d'un robot jouant d'un instrument de musique est un véritable défi lorsque l'on sait qu'un musicien met plusieurs années à maîtriser parfaitement son instrument.

Ce projet a été entrepris avec plusieurs objectifs :

— Participer à un concours international de robots jouant d'un instrument non modifié : l'« Artemis Orchestra Competition ». Cette participation s'est conclue par la victoire du robot clarinettiste en juin 2008. Il était en concurrence avec un robot guitariste et un robot violoniste.

— Intéresser de jeunes étudiants aux télécommunications par des présentations du robot dans les écoles.

— Pour le laboratoire d'acoustique musicale, ce robot représente un outil expérimental pour étudier la clarinette, le clarinettiste ainsi que leurs interactions mutuelles. Ce type d'étude permet l'élaboration de conseils techniques pour l'enseignement de la clarinette.

Ce robot, tout comme un vrai musicien, possède une « bouche » avec une pompe pour poumons ainsi qu'une lèvre et une langue mécaniques. Ses « doigts » sont des pistons et son « cerveau » est sous forme d'électronique embarquée.

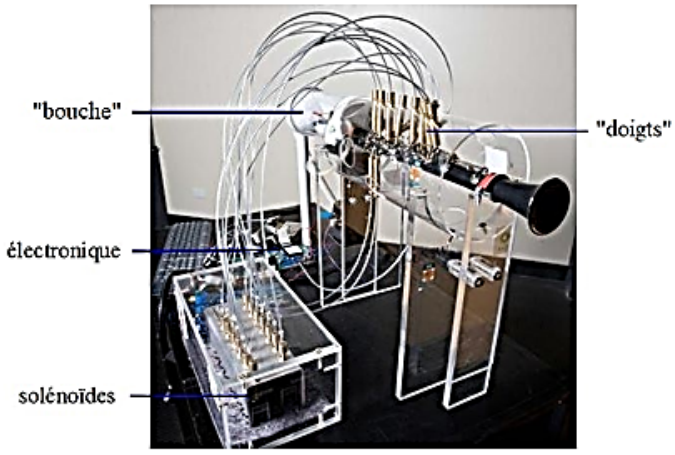


**Figure 1** Robot clarinettiste construit par le laboratoire d'acoustique musicale de l'UNSW et par l'institut de recherche en télécommunication NICTA.

Ce sujet propose l'étude de différents éléments constitutifs du robot, puis, dans un second temps, l'acquisition et l'analyse du son produit par la clarinette.

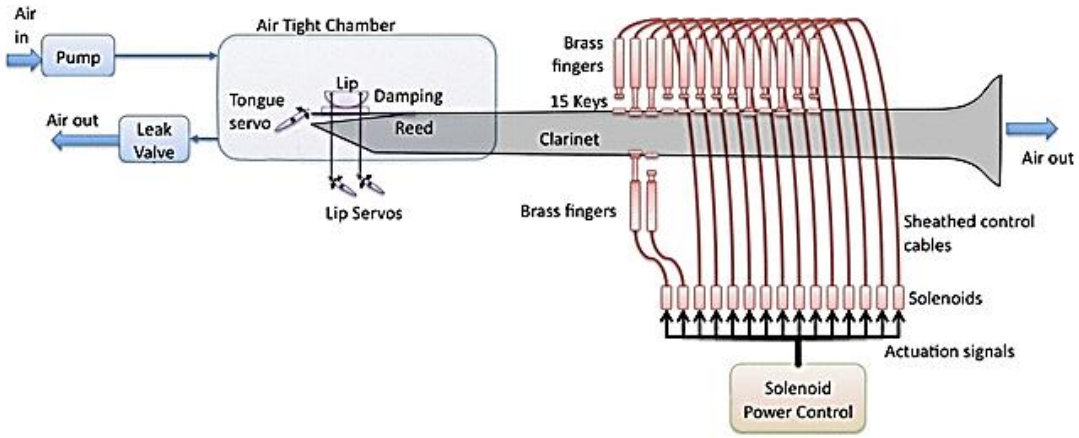


**Figure 2** **Schéma de principe du robot clarinettiste**, extrait de l'article d'André Almeida, « Clarinet parameter cartography : automatic mapping of the sound produced as a function of blowing pressure and reed force », Proceedings of the International Symposium on Music Acoustics, août 2010.

### *I.B – La « bouche » : contrôle de la pression*

La « bouche » du robot est constituée d'une enceinte dans laquelle la pression est régulée par un asservissement. Une pompe y injecte de l'air avec un débit réglable et une valve laisse sortir une quantité plus ou moins importante d'air de manière à respecter la commande. Envoyée par l'ordinateur au système de régulation de la pression, la commande est représentée par un nombre entier compris entre 0 et 1024. Un étalonnage a été réalisé pour connaître le lien entre la commande envoyée et la pression obtenue dans la « bouche ».

La méthode utilisée pour mesurer la pression dans la « bouche » est la suivante : on installe un tuyau entre la « bouche » et une colonne d'eau dans un tube en forme de U (figure 8) ; ce tube est solidaire d'un mètre gradué, qui permet de lire la différence de hauteur entre les deux parties rectilignes du tube. La différence de hauteur d'eau a été relevée pour différentes commandes envoyées.


**Figure 8** Colonne d'eau en U utilisée pour calibrer le capteur de pression.

**Q 19.** Déterminer sur combien de bits est codée la commande en pression.

### I.B.1) Lien entre la différence de hauteur d'eau et la pression

On considère un volume mésoscopique d'eau, compris entre les coordonnées cartésiennes $x$ et $x+\mathrm{d}x$, $y$ et $y+\mathrm{d}y$, $z$ et $z+\mathrm{d}z$. L'axe $Oz$ est orienté selon la verticale ascendante. On note $P$ la pression dans l'eau et on considère que celle-ci ne dépend que de la coordonnée $z$. L'eau est assimilée à un fluide incompressible.

MP

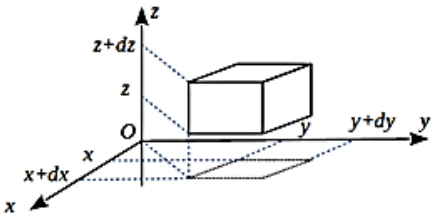


**Figure 9** Volume mésoscopique d'eau considéré.

**Q 20.** Exprimer la résultante des forces de pression $\overrightarrow{\mathrm{dF}}_p$ subie par le volume mésoscopique d'eau en fonction de la pression en $z$ et $z+\mathrm{d}z$ et d'une surface infinitésimale adaptée.

**Q 21.** On considère que ce volume d'eau est à l'équilibre mécanique dans le champ de pesanteur terrestre. Établir la relation fondamentale de la statique des fluides : $\frac{\mathrm{dP(z)}}{\mathrm{dz}} = -\rho_{\mathrm{eau}}\mathrm{g}$, où $\rho_{\mathrm{eau}}$ désigne la masse volumique de l'eau et $g$ la norme de l'accélération de la pesanteur.

**Q 22.** En déduire l'expression de la différence de pression $\Delta P$ entre la « bouche » et l'atmosphère en fonction de la masse volumique de l'eau $\rho_{\mathrm{eau}}$, de l'accélération de la pesanteur $g$ et la différence de hauteur d'eau $\Delta h$.

On obtient : $\Delta \boldsymbol{P} = \boldsymbol{\rho_{eau}}.\boldsymbol{g}.\Delta \boldsymbol{h}$

### I.B.2) Étalonnage : lien entre la commande envoyée au système de régulation de la pression dans la « bouche » du robot et la pression réelle obtenue

La différence de hauteur d'eau $\Delta h$ est relevée pour différentes commandes envoyées. Les mesures obtenues sont présentées dans la figure 10.

On choisit de modéliser, dans la plage testée, la relation entre la différence de hauteur $\Delta h$ et la commande envoyée, par une loi linéaire. Un programme Python, dont le code est donné dans la figure 11, est utilisé pour obtenir la valeur du coefficient directeur correspondant et son incertitude.

**Q 23.** Compte tenu du programme de la figure 11, indiquer quelle est la valeur de la précision considérée pour la lecture de la hauteur du niveau d'eau. On rappelle que pour obtenir une différence de hauteurs, 2 lectures de hauteur sont nécessaires. Justifier clairement en précisant quelle ligne du programme a été utilisée. Des rappels sur les incertitudes sont donnés en fin de sujet.

**Q 24.** Expliquer si ce programme prend en compte une incertitude pour la commande envoyée. Justifier.

**Q 25.** La ligne 5 comporte une erreur. Proposer une écriture correcte de la commande de la ligne 5. Compléter les lignes 18 et 21.

Le programme Python affiche le résultat suivant :

```
Coef directeur : a =  0.6475357649091059 +- 0.0004969890859212488
```

**Q 26.** Indiquer, en justifiant la réponse, l'unité du coefficient directeur.

| Commande | $\Delta h$ |
|---|---|
| 350 | 217 |
| 400 | 251 |
| 450 | 292 |
| 500 | 335 |
| 550 | 360 |
| 600 | 381 |
| 650 | 421 |
| 700 | 454 |
| 750 | 489 |

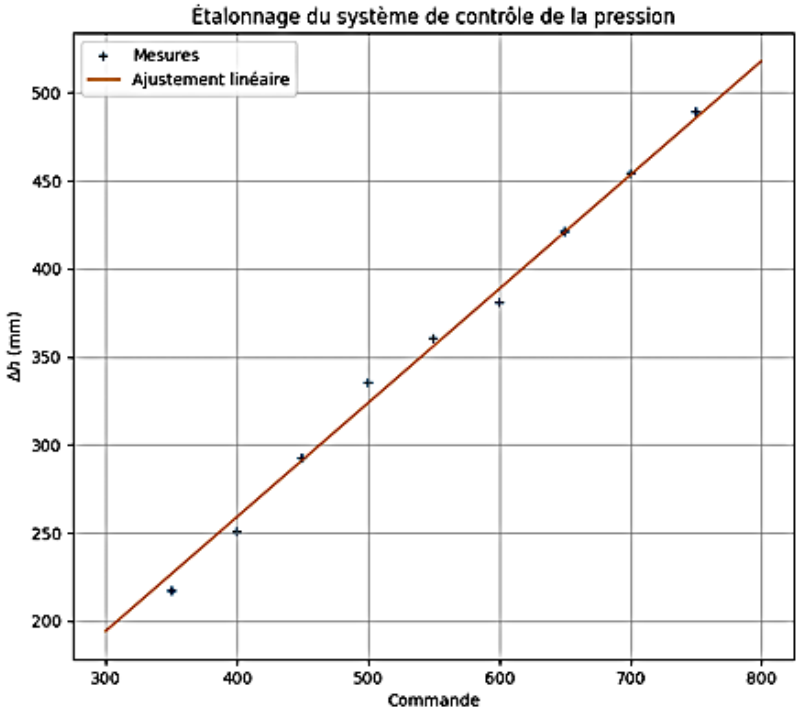


**Figure 10** Résultats expérimentaux des mesures de différence de hauteur d'eau pour différentes commandes envoyées et graphique avec une modélisation linéaire des données.

```
1   import numpy as np
2
3   # Valeurs expérimentales
4   # Commandes envoyées au système
5   x = np.array([350 + i*25 for i in range(9)])
6   nx = len(x) # nombre de de mesures
7   # Différences de hauteur (mm)
8   Dh = np.array([217, 251, 292, 335, 360, 381, 421, 454, 489])
9   # incertitude-type sur les hauteurs (mm)
10  uDh = (1/3**0.5)*2**0.5
11
12  # régression linéaire en imposant une ordonnée à l'origine nulle
13
14  # Simulation Monte-Carlo
15  N = 1000 # nombre d'expériences simulées
16  aMC = [] # Liste contenant les coefficients directeurs des expériences simulées
17
18  for i in range(...):
19      # Tirage au sort des pressions
20      Dh_MC = []
21      for j in range(...):
22          #P_MC.append(np.random.normal(P[j], uP))
23          Dh_MC.append(np.random.normal(Dh[j], uDh))
24      Dh_MC=np.array(Dh_MC)
25      # Calcul du coefficient de proportionnalité optimal avec cette série de
26                #valeurs tirées au sort
27      aMC.append(np.dot(x,Dh_MC) / np.dot(x,x))
28
29  # Analyse statistique des valeurs de a et de b obtenues
30  print("Coef directeur : a = ", np.mean(aMC), "+-", np.std(aMC, ddof = 1))
```

**Figure 11** Programme Python pour obtenir la valeur du coefficient directeur de la loi linéaire qui modélise la relation entre la différence de hauteur et la commande envoyée au système.

**Q 27.** Écrire la valeur du coefficient directeur avec son incertitude-type en conservant un seul chiffre significatif pour l'incertitude-type.

**Q 28.** En déduire la valeur de la commande à envoyer pour obtenir une différence de pression dont la valeur est la plus proche possible de $0,030$ bar.

---

# Données

| | |
|---|---|
| Constante d'Avogadro | : $N_A = 6,022\ 10^{23}\ \text{mol}^{-1}$ |
| Accélération de la pesanteur | : $g = 9,81\ \text{m} \cdot \text{s}^{-2}$ |
| Masse molaire du cuivre | : $M_{\text{Cu}} = 63,55\ \text{g} \cdot \text{mol}^{-1}$ |
| Masse molaire de Na(OH) | : $M_{\text{Na(OH)}} = 40,0\ \text{g} \cdot \text{mol}^{-1}$ |
| Masse volumique du cuivre | : $\rho_{\text{Cu}} = 8,96\ 10^3\ \text{kg} \cdot \text{m}^{-3}$ |
| Masse volumique de l'eau | : $\rho_{\text{eau}} = 1,00\text{e}3\ \text{kg} \cdot \text{m}^{-3}$ |
| Capacité thermique massique du cuivre | : $c_{\text{Cu}} = 380\ \text{J} \cdot \text{kg}^{-1} \cdot \text{K}^{-1}$ |
| Rayon atomique du cuivre | : $r_{\text{Cu}} = 128\ \text{pm}$ |
| Énergie d'activation de la polymérisation du MMA | : $E_a = 162\ \text{kJ} \cdot \text{mol}^{-1}$ |
| Produit ionique de l'eau à 25°C | : $K_e = 10^{-14}$ |
| $\text{pK}_\text{a}$ du couple $\text{NH}_4^+/\text{NH}_3$ | : $\text{pK}_\text{a} = 9,2$ |
| Produit de solubilité de $(\text{NH}_4)_2\,\text{SO}_4$ dans l'eau à 25°C | : $K_s((\text{NH}_4)_2\text{SO}_4) = 773$ |
| Enthalpie standard de dissolution des cristaux de sulfate d'ammonium à 298 K | : $11,1\ \text{kJ} \cdot \text{mol}^{-1}$ |
| Enthalpie standard de formation de la cyanhydrine d'acétone (espèce notée A sur la figure 13) | : $\Delta_f H^\circ = -120\ \text{kJ} \cdot \text{mol}^{-1}$ |
| Enthalpie standard de formation du méthacrylamide (espèce notée B sur la figure 13) | : $\Delta_f H^\circ = -470\ \text{kJ} \cdot \text{mol}^{-1}$ |
| Célérité du son dans l'air à 25°C | : $c_{\text{son}} = 346\ \text{m} \cdot \text{s}^{-1}$ |

# Formulaire

**Incertitudes-types** Notation : $u$ désigne l'incertitude-type.

— incertitude-type pour une mesure analogique : $u = \frac{\text{graduation}}{\sqrt{12}}$ ;

— incertitude-type pour une mesure de précision ou tolérance $p$ : $u = \frac{p}{\sqrt{3}}$.

**Relation de propagation des incertitudes**

— Pour une grandeur $G$ de la forme $G = k(\alpha a + \beta b)$ ($k$, $\alpha$ et $\beta$ sont des constantes), l'incertitude-type $u(G)$ est calculée ainsi : $u(G) = k\sqrt{\left(\alpha u(a)\right)^2 + \left(\beta u(b)\right)^2}$

— Pour une grandeur $G$ de la forme $G = ka^\alpha b^\beta$ ($k$, $\alpha$ et $\beta$ sont des constantes), l'incertitude-type $u(G)$ est calculée ainsi : $u(G) = G\sqrt{\left((\alpha\frac{u(a)}{a}\right)^2 + \left(\beta\frac{u(b)}{b}\right)^2}$.