# TP3 – Capacités numériques
## Simulations Monte Carlo
## Méthode d'Euler

**Outils et capacités numériques au programme :**

La prise en compte de capacités de codage en langage Python dans la formation des étudiants inclue l'utilisation de fonctions extraites de diverses bibliothèques. Elle vise à une meilleure appréhension des principes mis en œuvre par les différents logiciels de traitement des données dont l'utilisation est par ailleurs toujours recommandée. Elle a aussi pour objectif de mobiliser ces capacités dans un contexte concret, celui de la physique et de la chimie. Cette formation par le codage permet également de développer des capacités utiles à la physique-chimie comme le raisonnement, la logique ou la décomposition d'un problème complexe en étapes plus simples.

| | |
|---|---|
| **Outils graphiques**<br>Représentation graphique d'un nuage de points.<br>Représentation graphique d'une fonction. | Utiliser les fonctions de base de la bibliothèque matplotlib pour représenter un nuage de points.<br>Utiliser les fonctions de base de la bibliothèque matplotlib pour tracer la courbe représentative d'une fonction. |
| **Équations différentielles**<br>Équations différentielles d'ordre 1.<br>Équations différentielles d'ordre supérieur ou égal à 2 | Mettre en œuvre la méthode d'Euler explicite afin de résoudre une équation différentielle d'ordre 1.<br>Transformer une équation différentielle d'ordre n en un système différentiel de n équations d'ordre 1.<br>Utiliser la fonction **odeint** de la bibliothèque **scipy.integrate** (sa spécification étant fournie). |
| **Probabilités – statistiques**<br>Variable aléatoire.<br>Régression linéaire. | Utiliser les fonctions de base des bibliothèques **random** et/ou **numpy** (leurs spécifications étant fournies) pour réaliser des tirages d'une variable aléatoire.<br>Utiliser la fonction **random.normal** de la bibliothèque **numpy** (sa spécification étant fournie) pour simuler un processus aléatoire.<br>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.<br>Déterminer la moyenne et l'écart-type d'un ensemble de tirages d'une variable aléatoire.<br>Utiliser la fonction **polyfit** de la bibliothèque **numpy** (sa spécification étant fournie) pour exploiter des données. |
| **Incertitudes-types composées.** | Simuler, à l'aide d'un langage de programmation ou d'un tableur, un processus aléatoire permettant de caractériser la variabilité de la valeur d'une grandeur composée. |
| **Incertitudes sur les paramètres d'un modèle obtenu par régression linéaire.** | 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. |

**A faire pour le jeudi 24/09 : Lire le Complément de TP Mesures et incertitudes, le poly Outils Numérique Résolution numérique d'ED et ce document et répondre aux questions ✎.**
**Fichiers Python (à compléter) disponibles sur** https://cahier-de-prepa.fr/mp-lafayette/docs?rep=135

# A) Simulations Monte-Carlo

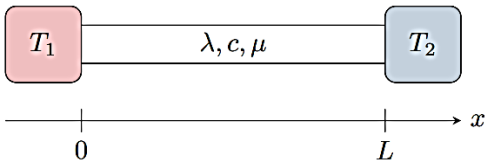
Lire Complément de TP Mesures et incertitudes § C et D

## 1) Incertitude-type composée

✐➲ 1. Répondre à la question du § D.2.b du Complément de TP Mesures et incertitudes (fichier python TP3_A.1 … disponible sur Cahier de Prépa).

## 2) Régression linéaire et incertitude-type

On considère une barre en aluminium, de conductivité thermique $\lambda = 191\ W.m^{-1}.K^{-1}$, de longueur $L$, de section rectangulaire $S = 2\ \text{cm}^2$, calorifugée latéralement, telle que chacune de ses extrémités est en contact avec un thermostat de température $T_i, i = 1,2$. 8 sondes de température sont disposées le long de la barre tous les $2{,}2\ cm$. La précision des sondes de température vaut $\Delta T = 0{,}6\ °C$.




En régime stationnaire, on obtient les résultats suivants :

| x (m) | T(x,tf) (°C) |
|---|---|
| 0 | 61,5 |
| $2{,}20.10^{-2}$ | 55,8 |
| $4{,}40.10^{-2}$ | 49,6 |
| $6{,}60.10^{-2}$ | 43,7 |
| $8{,}80.10^{-2}$ | 38 |
| $1{,}10.10^{-1}$ | 32,3 |
| $1{,}32.10^{-1}$ | 26,8 |
| $1{,}54.10^{-1}$ | 21,4 |

➲ 2. A l'aide du programme Python TP3_A.2 … disponible sur Cahier de Prépa, tracer le profil de température dans la barre en régime stationnaire.
➲ 3. Commenter cette courbe.
➲ 4. Compléter le programme Python pour modéliser le profil de température dans la barre. En déduire la valeur du flux thermique dans la barre.
➲ 5. Terminer le programme par une simulation Monte-Carlo* pour évaluer l'incertitude sur la pente de la régression linéaire.

**Données** :
`M=np.average(M_lst)` permet de calculer la **moyenne** des mesures de la liste M_lst.
`S=np.std(M_lst,ddof=1)` permet de calculer **l'écart type** des mesures de la liste M_lst.
`D=np.polyfit(x,y,1)` permet de déterminer la **pente** et **l'ordonnée à l'origine** de la **droite** (polynôme de degré 1) passant au plus près des points de coordonnées (x,y). D[0] donne la pente et D[1] donne l'ordonnée à l'origine.

* Inspirez-vous du programme donné dans le § D.3.b du Complément de TP Mesures et incertitudes (fichier Python Ct TP D.3.b…. disponible sur Cahier de Prépa).

# B) Résolution d'équations différentielles

Lire ChON1 Résolution numérique d'équations différentielles § A et B

**Données** :

On peut utiliser la méthode de résolution d'équations différentielles contenue dans la bibliothèque **scipy.integrate** : **odeint**.

La fonction **odeint** impose d'écrire l'équation différentielle sous la forme : $y'(t) = \frac{dy}{dt} = F(y,t)$

Les arguments de **odeint** sont :

- la fonction $F$ (dont les arguments sont $y$ et $t$, à donner dans cet ordre lors de sa définition),
- la condition initiale $y(t_0) = y_0$,
- `time` le tableau des $n$ instants (on cherche à résoudre l'équation différentielle en ces points).

NB : On peut utiliser la fonction odeint pour résoudre une ED d'ordre $p$.

Les arguments de **odeint** sont :

- la fonction $F$ dont les arguments sont $y$ et $t$ avec $y$ un tableau contenant $p$ éléments, la fonction $F$ retourne un tableau contenant $p$ éléments relatif au système différentiel ;
- les conditions initiales : tableau de $p$ éléments ;
- `time` le tableau des $n$ instants.

Pour une ED d'ordre $p$, la fonction **odeint** renvoie un tableau contenant $n$ tableaux à $p$ éléments.

## 1) Exemple d'équation différentielle du 1er ordre : champ de pression dans la troposphère

✎ ➲ 6. Répondre aux questions de l'exercice ci-dessous (fichier python TP3_B.1… à compléter disponible sur Cahier de Prépa).

Dans la troposphère, couche occupant les 11 premiers kilomètres de l'atmosphère, on peut montrer que le champ de pression $P(z)$ varie avec l'altitude $z$ selon l'équation différentielle

$$\frac{\mathrm{d}P}{\mathrm{d}z} = -\frac{Mg}{R(T_0 - \alpha z)}P$$

avec $M = 29 \cdot 10^{-3}\,\mathrm{kg \cdot mol^{-1}}$ la masse molaire de l'air, $g = 9{,}81\,\mathrm{m \cdot s^{-2}}$ l'accélération de la pesanteur, $R = 8{,}3\,\mathrm{J \cdot mol^{-1} \cdot K^{-1}}$ la constante des gaz parfaits, $T_0 = 288\,\mathrm{K}$ la température moyenne à la surface de la Terre et $\alpha = 6 \cdot 10^{-3}\,\mathrm{K \cdot m^{-1}}$ le gradient thermique.

**1** - Écrire la relation de récurrence issue du schéma d'Euler explicite appliqué à cette équation différentielle.

**2** - L'implémenter en Python pour trouver le champ de pression de 0 à 11 km d'altitude avec un pas de 1 m.

**3** – Résoudre avec la fonction **odeint** de la bibliothèque **scipy.integrate**. Commenter.

*Rq pour les MP : on démontrera cette ED au ChT5.*

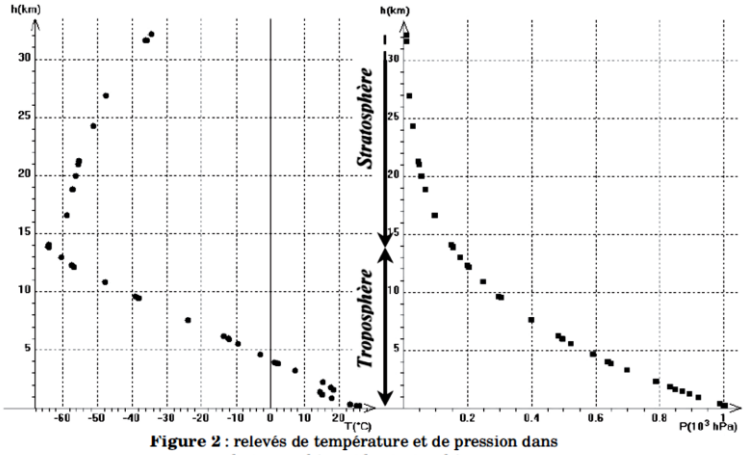



**Figure 2** : relevés de température et de pression dans la troposphère et la stratosphère

## 2) Exemple d'équation différentielle du 2e ordre : pendule pesant



➲ 7. Répondre aux questions de l'exercice ci-dessous (fichier python TP3_B.2… à compléter disponible sur Cahier de Prépa).

On étudie ici le mouvement du pendulier d'une horloge, cf figure ci-contre. Il s'agit d'un solide de centre d'inertie $G$, de masse $m = 2\ kg$, en rotation autour de l'axe horizontal $(Ox)$ fixe par rapport au référentiel terrestre. La rotation du pendulier est assurée par une liaison pivot parfaite d'axe $(Ox)$.

La position du pendulier est repérée par l'angle $\theta$ formé entre l'axe vertical descendant $(Oz)$ et la droite $(OG)$. Le moment d'inertie du pendule par rapport à l'axe $(Ox)$ est noté $J_{(Ox)}$ et la distance $OG$ est notée $d$ : $J_{(Ox)} = 2\ m^2.kg$ et $d = 1\ m$.

On ne s'intéressera pas au mécanisme entretenant les oscillations et on négligera les frottements dus à l'air.

1) Faire un schéma du dispositif étudié : représenter les axes $(Ox)$ et $(Oz)$, le centre d'inertie $G$ et l'angle $\theta$.
2) Montrer que l'angle $\theta$ vérifie l'équation différentielle suivante :

$$\frac{d^2\theta}{dt^2} + \frac{mgd}{J_{(Ox)}}\sin(\theta) = 0$$

3) Ecrire les relations de récurrence issues du schéma d'Euler explicite appliqué à cette équation différentielle.
4) Analyser l'équation différentielle pour proposer une valeur de pas de temps raisonnable a priori.
5) L'implémenter en Python pour résoudre l'équation sur une durée totale correspondant à 3 oscillations. On supposera le pendulier lâché sans vitesse initiale depuis un angle de 45°.
6) Vérifier qu'en augmentant la durée totale, la solution numérique finit toujours par diverger, et ce quel que soit le pas de temps.
7) Résoudre avec la fonction **odeint** de la bibliothèque **scipy.integrate**. Vérifier que le problème soulevé à la question précédente n'a plus lieu avec **odeint**.
8) Pour différentes conditions initiales (vitesse initiale nulle et différents angles initiaux), vérifier qu'il y a perte d'isochronisme des oscillations (i.e. que pour des oscillations d'amplitude élevée, la période dépend de l'amplitude). Il s'agit d'un effet de la non-linéarité de l'équation.