TP3 – Capacités numériques
Simulations Monte Carlo
Méthode d'Euler

<u>Outils et capacités numériques au programme :</u>
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 Représentation graphique d'un nuage de points. Représentation graphique d'une fonction.	Utiliser les fonctions de base de la bibliothèque matplotlib pour représenter un nuage de points. Utiliser les fonctions de base de la bibliothèque matplotlib pour tracer la courbe représentative d'une fonction.
Équations différentielles Équations différentielles d'ordre 1. 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. Transformer une équation différentielle d'ordre n en un système différentiel de n équations d'ordre 1. Utiliser la fonction **odeint** de la bibliothèque **scipy.integrate** (sa spécification étant fournie).
Circuit linéaire du 1er et du 2e ordres	Mettre en œuvre la méthode d'Euler à l'aide d'un langage de programmation pour simuler la réponse d'un système linéaire du 1er / 2e ordre à une excitation de forme quelconque.
Petits mouvements au voisinage d'une position d'équilibre stable, approximation locale par un puits de potentiel harmonique.	A l'aide d'un langage de programmation, résoudre numériquement une équation différentielle du deuxième ordre nonlinéaire et faire apparaître l'effet des termes non-linéaires.
Probabilités – statistiques Variable aléatoire. 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. 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. Déterminer la moyenne et l'écart-type 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.
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.
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.

<u>A faire pour le jeudi 25/09 :</u> Lire le Complément de TP Mesures et incertitudes, le ChON1 et ce document et répondre aux questions ✎.
Fichiers Python (à compléter) disponibles sur https://cahier-de-prepa.fr/mp-lafayette/docs?rep=115

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 à compléter disponible sur Cahier de Prépa).

2) Régression linéaire

Au TP4C, on étudiera une barre en aluminium de section rectangulaire $S = 2\ \mathrm{cm}^2$ chauffée par effet Joule à l'aide d'une résistance à l'une de ses extrémités. Une circulation d'eau froide permet le maintien de la température à l'autre extrémité de la barre à une valeur sensiblement constante.

L'acquisition est réalisée avec un logiciel via 8 sondes de température disposées sur la barre tous les $2{,}2\ cm$.

La puissance de chauffe est choisie à $P = 10\ W$.

Données : Pour la barre en aluminium de ce dispositif, on attend une conductivité thermique de $191\ W.m^{-1}.K^{-1}$ et un coefficient de diffusion $D \approx 1.10^{-4}\ m^2.s^{-1}$.

En régime (quasi-)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
$1{,}10.10^{-2}$	32,3
$1{,}32.10^{-2}$	26,8
$1{,}54.10^{-2}$	21,4

➲ 2. A l'aide d'un programme Python, tracer le profil de température dans la barre en régime stationnaire.

➲ 3. Commenter cette courbe. Comment se simplifie l'équation de la diffusion thermique dans le cas stationnaire ? Que peut-on dire du flux thermique $\Phi(x)$ au travers d'une section S de la barre ? Que vaut-il ?

➲ 4. Compléter votre programme Python, en utilisant la fonction polyfit* de la bibliothèque numpy pour modéliser le profil de température dans la barre. En déduire la conductivité thermique de la barre d'aluminium. Commenter.

➲ 5. La demi-largeur de l'intervalle de valeurs acceptables pour les sondes de température vaut $\Delta T = 0{,}6\ °C$. En utilisant une simulation Monte-Carlo sous Python**, évaluer l'incertitude sur la pente de la régression linéaire et en déduire l'incertitude sur la conductivité thermique.

* `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 / adaptez le programme donné dans le § D.3.b du Complément de TP Mesures et incertitudes (fichier python correspondant 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 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.

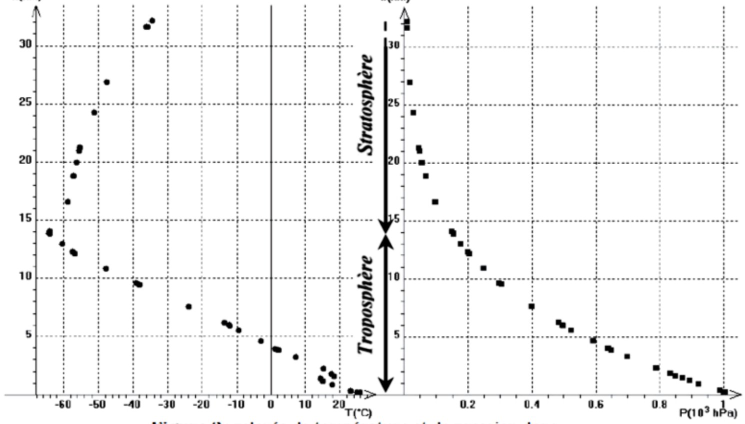

Figure 2 : relevés de température et de pression dans la troposphère et la stratosphère

2) Exemple d'équation différentielle du 2^e ordre : pendule simple

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

L'angle formé par un pendule simple de longueur $\ell = 20\,\text{cm}$ avec la verticale vérifie l'équation différentielle

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{g}{\ell}\sin\theta = 0\,.$$

1 - Écrire les relations de récurrence issues du schéma d'Euler explicite appliqué à cette équation différentielle.

2 - Analyser l'équation différentielle pour proposer une valeur de pas de temps raisonnable a priori.

3 - L'implémenter en Python pour résoudre l'équation sur une durée totale correspondant à trois oscillations. On supposera le pendule lâché sans vitesse initiale depuis un angle de 45°.

4 - Vérifier que, en augmentant la durée totale, la solution numérique finit toujours par diverger, et ce quel que soit le pas de temps

5 – 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**.

6 – 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.