Python (FIP, L3), travaux dirigés

Pierre Cladé, Cyrille Solaro, Kévin Falque

2023/2024

L'ensemble des liens et resources pour ce cours se trouveront ici : github.com/clade/InfoFIP2023.

1 Géneralités sur Python

Cette partie contient des exercices sur des notions générales de Python

1.1 Jupyter notebook

- Créer un répertoire pour le cours de cette année et un sour répertoire pour le TD d'aujourd'hui (nommé par exemple TD1 ou yyyy_mm_dd, pour l'année, le mois et le jour). Retrouver ce repertoire dans l'arborescence de votre système de fichier.
- Dans ce répertoire, créer un notebook.
- Si vous n'êtes par famillier avec les notebooks, voici quelques astuces :
  - Utilisation de la touche 'Tab' pour la complétion automatique
  - Utiliser la combinaison Ctrl + Entrée pour valider une cellule
  - Menu : Cellule -> type de cellule -> markdown pour créer une zone de texte

Par exemple, créer au début de votre notebook les blocs suivants (zone de texte et calcul). Il est possible de rentrer des équations en utilisant le format latex : `$E=mc^2$`

Le format utilisé est le markdown fr.wikipedia.org/wiki/Markdown.

Relativité

Tout le monde connait la formule $E = mc^2$. Mais que vaut exactement l'énergie de masse d'une pomme de 200g ?

```
masse_d_une_pomme = 0.2
c = 3E8

energie_masse_pomme = masse_d_une_pomme*c**2
print(f"L'énergie de masse d'une pomme est {energie_masse_pomme:.3e} J")
```

1.2 Fichiers avec Python et les notebook

- A l'aide de Python, créer un fichier contenant le mot "Bonjour"
- Ouvrir ce fichier à l'aide d'un éditeur de texte et modifier le.
- Lire à nouveau le fichier avec Python

1.3 Analyse des données du CAC 40

L'objectif de cet exercice est de manipuler des données à l'aide de listes et de dictionnaires (sans utiliser de tableaux numpy).

Télécharger depuis le site https://fr.finance.yahoo.com/quote/%5EFCHI/history/ les données du CAC 40 depuis 3 ans. Le fichier obtenu contient : la date, le cours d'ouverture, le plus haut, le plus bas, le cours de clôture, le cours de clôture ajusté et le volume de titres échangés. Ce contenu est séparé par des `,`.

Remarque : la méthode `split` d'une chaine de caractère permet de la séparer en plusieurs sous chaînes. Par exemple :

```
s = '12,bonjour,3.14'
print(s.split(','))
```

On suprimera la première ligne du fichier.

1. Afficher les 10 premières lignes du fichier
2. Créer une liste contenant le plus haut de la bourse jour par jour. Qu'elle est la valeur la plus haute du CAC 40 ?
3. Écrire une fonction qui prend le jour, le mois et l'année et renvoie une chaîne de caractère pour la date sous la forme : aaaa-mm-jj (par exemple 1515-09-13)
4. Écrire une fonction qui renvoie le plus bas d'une journée à partir du jour, du mois et de l'année. Cette fonction renverra une exception si le jour n'existe pas.
5. Plutôt qu'une liste, on va utiliser un dictionnaire dont la clé sera la date. Créer un dictionnaire qui contiendra pour chaque jour un dictionnaire avec pour clé : 'haut', 'bas', 'ouverture', 'fermeture', 'volume'. Par exemple on pourra utiliser : `cac_40['2020-10-05']['haut']`
6. Enregistrer ce dictionnaire dans un fichier au format json. Vérifier que vous pouvez l'ouvrir !
7. Regarder la documentation de la fonction `parse` du module `dateutil.parser` et trouver le moyen de convertir une date sous la forme `2020-10-05` en un nombre (nombre de jour ou nombre de seconde depuis une date donnée).
8. Tracer l'évolution au cours du temps de CAC 40.

2 Numpy

2.1 Formule de Simpson

On rappelle la formule de Simpson pour le calcul approché d'une intégrale :

$$\int_a^b f(x)\mathrm{d}x \approx \Delta_x \sum_{i=0}^{N-1} \frac{f(x_i) + 4f(x_i + \frac{\Delta_x}{2}) + f(x_i + \Delta_x)}{6} \equiv I(f; a, b, N)$$

où $\Delta_x = \frac{b-a}{N}$ et $x_i = a + i\Delta_x$.

1. Ecrivez une fonction `simpson_slow` qui calcule l'intégrale d'une function f entre a et b avec N pas avec la méthode de Simpson en utilisant une boucle (for loop).
2. Ecrivez une autre fonction `simpson_fast` qui fait la même chose sans utiliser de boucle (on suposera que la fonction f est vectorisée).
3. Calculez l'intégrale de $f(x) = \frac{1}{1+x^2}$ entre 0 et 1 pour $N = 1000$ et comparez le temps entre les deux fonctions en ecrirant `%timeit` avant la commande.
4. Calculez la valeur théorique $I^\star$ de l'intégrale et tracez en échelle logarithmique le residu $|I^\star - I(f, 0, 1, N)|$ par rapport à N. Qu'elle est la vitesse de convergence de cette intégrale ?

La solution analytique de l'intégrale est : $\int_0^1 \frac{dx}{1+x^2} = \arctan(x)|_0^1 = \pi/4$

2.2 Volume d'une sphère

On considère un nuage de points (x, y) dans un plan 2D. Les variables x et y sont indépendantes et uniformément reparties entre -1 et 1.

1. En utilisant la fonction `np.random.rand`, créer un nuage de M points et tracer ce nuage. On pourra prendre $M = 1000$
2. Tracer dans une autre couleur les points dans un cercle de rayon 1.
3. En prenant M assez grand (par exemple 10^8), calculer la probabilité d'être dans le cercle. En déduire une estimation de la surface d'un disque de rayon 1.
4. Même question que la question 3, mais dans un espace de dimension N. Par exemple $N = 5$. On écrira une fonction.

2.3 Statistiques sur le COVID-19

Extrait du sujet de 2020/2021

Le fichier `data_exo_1.dat` contient trois colonnes : la première est la date, la seconde le nombre cumulé de décès en hôpital liés au COVID-19 et la troisième le nombre cumulé de décès en EHPAD liés au COVID-19.

Le fichier commence le lundi 2 mars 2020. C'est l'origine de nos dates (jour 0). Dans la suite, on entend par date ou numéro du jour le nombre de jours écoulés depuis le 2 mars 2020. Ce fichier a été extrait de la base de donnée en janvier 2021 (il ne comprend donc que la première et début de la deuxième vague).

1. Lire ce fichier à l'aide de la commande loadtxt, et extraire les trois colones dans trois variables.
2. Tracer le graphe du nombre de décès cumulé en hôpital et en EHPAD en fonction de la date.
3. Ajouter sur le même graph le nombre total de décés cumulé
4. Au cours des 30 premiers jours, le nombre cumulé de décès en hôpital suit une loi proche d'une exponentielle. Tracer cette courbe en échelle semi-logarithmique.
5. Au cours de ces 30 premiers jours, le nombre de décès peut s'écrire $N(j) = N_0(1+a)^{j-j_0}$. Tracer cette courbe avec les paramètres pour lesquels elle s'ajuste bien à l'œil. Quelle est la valeur de a? (Nous ne demandons pas dans cette question de faire un ajustement, mais simplement de superposer les deux courbes).
6. Calculer le tableau du nombre quotidien de décès en hôpital et en EHPAD en fonction de la date. Et tracer la courbe pour l'hôpital. (Remarque, il est possible de faire cette opération sans boucle)
7. Quel jour a connu le plus grand nombre de décès en hôpital ? (on utilisera la fonction `argmax`)
8. Combien de jours ont connu plus de 50 décés en EHPAD ?

9. Comment extraire en une ligne le nombre de décés pour un jour de la semaine donné ?
10. Calculer et tracer le nombre total de décès en hôpital par jour de la semaine. Pour quel jour de la semaine enregistre-t-on le moins de décès ?

2.4 Analyse de données de gravimètres.

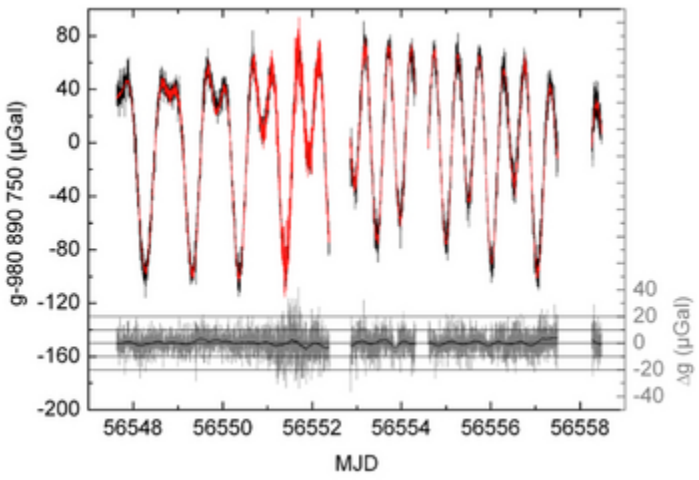
Extrait du sujet d'examen posé en 2021/2022

Un gravimètre est un appareil qui mesure l'accélération de la pesanteur. A Paris, elle vaut $g \sim 9.81$ m/s^2. Dans cet exercice nous allons analyser des données provenant du gravimètre à atomes froids du laboratoire LNE-SYRTE (à Paris) et celles provenant d'un gravimètre à supraconducteur qui ont été enregistrées en parallèle, pendant 27 jours, entre le 7 avril et le 4 mai 2015. Le gravimètre à atomes froids (CAG, cold atoms gravimeter) est un gravimètre dit absolu dans le sens où il mesure g directement en m/s^2. Le gravimètre à supraconducteur ('iGRAV') est un gravimètre dit relatif dans le sens où il mesure une tension en V qui peut être reliée à g. Ce dernier doit donc être calibré. Ceci peut justement se faire à l'aide du CAG. C'est ce que nous allons faire dans un premier temps.

Vous trouverez un fichier texte `data/CAGiGrav.txt` dans le dossier data qui contient : la date (en jour Julien modifié 'MJD'), la mesure du gravimètre (en nm/s^2, par rapport à la valeur $g = 9.808\ 907\ 500$ m/s^2), la mesure de l'iGRAV (en V) et le résidu entre les deux mesures après étalonnage de l'iGRAV (en nm/s^2). Ce contenu est séparé par des `'tabulations'`. La première ligne correspond à l'en-tête.

1. Importer le fichier `'CAGiGRAV.txt'` et créer quatres tableaux de flottants (`float`) correspondant à la date, la mesure du CAG, celle de l'iGRAV et le résidu après calibration. On pourra utiliser la fonction `np.loadtxt()` dont les arguments peuvent être retrouvés à partir de sa documentation : il faudra savoir 1/ comment utiliser le `\t` comme séparateur et 2/ comment éliminer la première ligne (en-tête) du fichier.
2. Tracer l'évolution de l'accélération de la pesanteur en fonction de la date (en MJD) mesurée à l'aide du CAG et à l'aide du iGRAV. On pourra utiliser la fonction `plt.subplots` dont les arguments peuvent être retrouvés à partir de sa documentation : notamment un des arguments permettra de partager l'axe des abscisses.

Le résultat typique que l'on observe est le suivant:



Les variations de grandes amplitudes et de longues périodes sont liées à l'attraction de la lune et du soleil (phénomène des marées). Dans ce graphique, les unités sont en Gal. 1 µGal = 1 nm/s^2.

Comme pour beaucoup de jeux de données expérimentales, il manque des données (ici soit parce que les mesures ont été perturbées par des tremblements de terre soit parce que l'asservissement des lasers (servant au refroidissement laser des atomes) du gravimètre atomique a sauté). Ces données manquantes sont représentées par des 'nan' (not a number).

La fonction `np.isnan` permet de créer un tableau de booléen contenant les nombres qui sont des 'nan'. Le code suivant permet de creer un masque (tableau de booléen) contenant uniquement les données valides en même temps pour les

deux gravimètres:

```
mask = ~(np.isnan(CAG) | np.isnan(iGRAV))
```

3. Pourquoi avoir utilisé ~ et | au lieu de `not` et `or` ?
4. Quelle était la valeur moyenne et l'écart type de g entre le 7 avril et le 4 mai 2015 (c'est à dire sur l'ensemble des données) ?

Les données de l'iGRAV sont données en volt. On note K le coeficient de calibration de l'instrument (en nm.s^{-2}/V). On prendra $K = -898.25$nm.s^{-2}/V.

5. Calculer et tracer la différence entre les mesures enregistrées par le CAG et celles enregistrées par l'iGRAV après calibration: CAG-K*iGRAV. Vérifier graphiquement que votre résultat correspond bien à la quatrième colonne du fichier `'CAGiGRAV.txt'`.
6. Quel est l'écart type de ces résidus sur la période des 27 jours de mesures ? Comparer celui-ci avec l'écart type des mesures du gravimètre atomique (CAG) seules, calculées à la question 4. Commenter.

Grâce au gravimètre à supraconducteur, il a été possible d'éliminer les variations dues aux effets des marées et d'en déduire le bruit du CAG (le bruit de l'iGRAV est négligeable en comparaison). Lorsque l'on n'a pas cet appareil (l'IGRAV), il est possible de caractériser le bruit de l'instrument (le CAG) grace à un outils mathématique appelé **la variance d'Allan**. L'idée est que entre deux mesures consécutives, le signal varie peu et que seul le bruit de l'instrument va changer le résultat.

La variance d'Allan est définie par :

$$\sigma^2_{\text{Allan}} = \frac{1}{2}\left\langle (x_{l+1} - x_l)^2 \right\rangle_l$$

7. (facultative) Question de statistique: vérifier que si les x_i sont indépendants, σ_{Allan} correspond à l'écart type des x.
8. Écrire une fonction `allan_variance(x)` qui calcule la variance d'Allan d'un jeu de données x. Bien que ce ne soit pas exactement le cas, nous allons ici supposer que chaque mesure du CAG est réalisée pendant un temps $\tau_0 = 177$ s. Calculer **l'écart type d'Allan** $\sigma_{Allan}(\tau_0)$ du jeu de données du CAG et vérifier que l'on a le même ordre de grandeur qu'à la question 6.
8. (bis) Si ce n'est pas déjà fait, écrire la fonction `allan_variance(x)` sans boucle for.

2.5 Vigicrue

Le réseau de prévision des crues, dénommé Vigicrues, publie sur internet le relevé des stations situés le long de fleuves et rivières de France.

Les données sont fournies au format JSON. Nous avons téléchargé un fichier qui contient des données d'une station située en Indre-et-loire `data/vigicrue.json`. Ce fichier contient un dictionnaire

1. Lire le fichier. Quels sont les clés de ce dictionnaire ?
2. On s'interesse au contenu "Serie" de ce dictionnaire qui est aussi un dictionnaire. La clé "LbStationHydro" contient la commune où la mesure à été prise. Donner le nom de cette commune ?

Les données qui nous interessent sont repérées par le clé "ObssHydro". La variable "DtObsHydro" contient le temps UNIX en milisecondes (temps depuis le 1 janvier 1970). La variable "ResObsHydro" la hauteur par rapport à une référence en m. Le jeu de donnée commence vers le 12 avril 2023 (temps unix de 1681257600 s)

3. Extraire le temps et l'enregistrer en heure par rapport au 12 avril 2023.
4. Tracer la hauteur en fonction du temps par rapport à l'instant présent
5. A quel jour correspond le maximum sur cette période ?

2.6 Loi de poisson et générateur de nombre aléatoire

Cet exercice s'appuie sur un sujet de l'IPT, avec un méthode légèrement différente. Regarder : Emergent Scientist 1, 7 (2017)

Une source lumineuse illumine un photomultiplicateur. Ce dispositif envoie un pulse digital d'environ 20 ns à chaque photon qu'il détecte. La sortie du photomultiplicateur est connectée à un dispositif informatique qui permet de compter le nombre de pulses reçu pendant une durée determinée.

Le nombre de photons qui arrive pendant une durée donnée suit une loi de Poisson, c'est à dire que la probabilité de détecter k photons est donnée par :

$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

où λ est le nombre moyen de photons. Le paramètre λ sera proportionnel à la durée t_0 pendant laquelle on mesure le nombre de photons : $\lambda = \Gamma t_0$.

On rappelle que l'écart type de la loi de Poisson vaut $\sqrt{\lambda}$

1. Le fichier de données est enregistré sous forme d'un fichier texte. Chaque point correspond à une mesure de durée $t_0 = 200\mu$s. Lire le fichier sous forme d'un tableau d'entier:

   Quel est le nombre moyen de photons reçu par seconde ?

2. Calculer l'écart type et vérifiez qu'il vaut $\sqrt{\lambda}$.

3. En utilisant la fonction `numpy.unique`, avec l'option `return_counts=True`, tracez la distribution de probabilité (créer un histogramme).

4. Tracez les points représentants $p(k)/p(k+1)$.

5. On peut créer un générateur de bits aléatoires à partir de cette séquence : si $x_{2j} > x_{2j+1}$ alors on prend 1, si $x_{2j} < x_{2j+1}$ on prend 0, sinon on élimine le point j.

   Créer une fonction `bits_aleatoires(data)` qui engendrer cette suite de bits aléatoire que l'on appellera a_j - une fois avec et sans de boucle `for`. Comparez les temps.

6. On peut ensuite créer une suite de nombre aléatoire $\{b_j\}$ entre 0 et 1 en regroupant les bits $\{a_i\}$ dans une manière que b_j soit écrit en binaire comme $(a_{Nj}, a_{Nj+1}, ..., a_{N(j+1)-1})$. On prendra par exemple N=11.

   $$b_j = \sum_{i=0}^{i=N-1} \frac{a_{Nj+i}}{2^{i+1}} \in [0, 1]$$

   Si $n = |\{a_j\}|$, le nombre d'elements dans la suite $\{a_j\}$, n'est pas divisible par N, supprimez les premiers $n\%N$ elements.

7. Si X et Y sont deux variables aléatoires ayant une distribution uniforme entre 0 et 1, alors on a

   $$P(X^2 + Y^2 < 1) = \frac{\pi}{4}$$

   Déterminez π en utilisant notre générateur de nombre aléatoire.

3 Ajustements de courbes et statistiques

3.1 Corrélation

On simule une jeu de données à l'aide du code suivant :

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0) # pour que le générateur "aléatoire" soit
                  # le même pour tout le monde
N = 100
x = np.linspace(2000, 2018, N)
y = np.arange(N)*0.2 + 45 + np.random.normal(size=N)

plt.plot(x, y, 'o');
```

1. Tracez et ajustez les données par une une droite $y = ax + b$.
2. Quel est l'incertitude sur b? Qu'en pensez-vous ?
3. Calculez la valeur et l'incertitude de votre fit en $x = 2010$.
4. Réalisez M simulations (M = 1000 par exemples); tracez sur un graph les coefficients a et b; Calculez la matrice de covariance.
5. Trouvez une fonction de fit plus pertinente pour ce problème.

3.2 Fit d'une image

Le fichier `data/double_star.txt` contient une image de 64 par 64 pixels d'une étoile double. L'objectif de cette partie est d'ajuster cette image par la somme de deux Gaussiennes afin de déterminer la distance entre les étoiles.

1. Charger et afficher le fichier à l'aide de la fonction imshow
2. Définir une fonction de fit et tracer une image qui ressemble à celle ci. On pourra utiliser la fonction `np.meshgrid`.

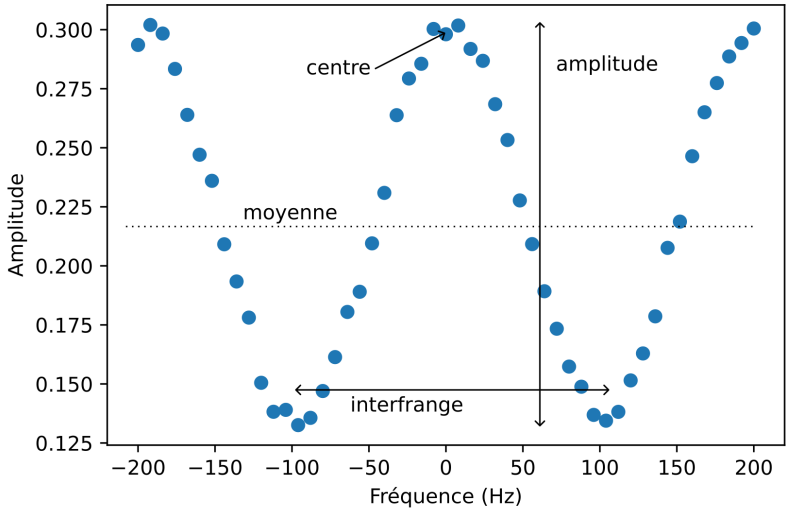
Il n'est pas possible de travailler directement sur des tableaux 2D pour le fit : l'ensemble des pixels de l'image doit être sous forme d'un tableau (taille N) ainsi que l'ensemble de coordonnées (x, y) des ces points (tailles $N \times 2$).

3. Ecrire une fonction qui s'adapte au contraite du fit et effectuer le fit.
4. Quelle est la distance et son incertitude (en pixel) entre les deux étoiles ?

3.3 Fit de franges d'interférence

On souhaite ajuster les franges d'un interféromètre atomique. Les données sont dans le fichier `data/fit_sinus.dat`. La première colonne du fichiers (axe x) représente une fréquence en Hz. La seconde colonne représente la population mesurée pour une fréquence donnée. L'objectif est de trouver la position de la frange centrale.

Voici à quoi ressemblent les données :



On ajustera par une fonction cosinus avec une amplitude, une moyenne, une position de la frange centrale et une largeur ajustable (interfrange), soit en tout quatre paramètres.

1. Lire le fichier et tracer les données.
2. Écrire la fonction de fit qui dépend des paramètres ci dessus. On appellera frange(x, ...). Tracez la courbe pour x entre $\pm$220 Hz avec des paramètres raisonables. On prendra un inter-frange de 200 Hz.
3. Calculer les paramètres optimaux. Quelle est la position la frange centrale ? Représentez les points et la courbe.
4. Quelle est l'incertitude sur la position de la frange centrale ?

4 Equations différentielles

4.1 Le pendule

On considère l'équation : $$\theta'' = -\sin\theta$$

Pour résoudre cette équation, on définit le tableau $y(t) = (\theta(t), \theta'(t))$.

1. Écrire la fonction Python `f(t, y)` qui renvoie la dérivée de y
2. Résoudre et tracer le resultat de l'équation différentielle pour les conditions initiales : $\theta(0) = 2\pi/4$ et $\theta'(0) = 0$
3. Vérifier que l'énergie totale est conservée

4.2 Equation de Schrödinger et états comprimés

Ce TD a pour but de simuler numériquement le TD "Etats quantiques d'atomes de césium dans un piège harmonique" du cours de physique quantique.

On prendra $m = 1$, $\hbar = 1$. On considère de plus un piège harmonique de pulsation $\omega = 2\pi$.

1. Ecrire (avec un papier et un crayon!) l'équation de Schrödinger.
2. Ecrire (avec un papier et un crayon) la fonction d'onde de l'état fondamental du piège harmonique.

On va échantillonner l'espace entre -5 et 5 avec un pas de 0.02

3. Ecrire une fonction Python qui calcule un état Gaussien d'écart type σ_x de vitesse v_0 et centré en x_0.
4. Ecrire une fonction Python qui calcule le Laplacien d'une fonction d'onde ψ.
5. Ecrire la fonction Python `psi_prime(t, psi, V)` qui donne la dérivée de psi. Vérifier graphiquement que l'on a bien $\frac{\partial \psi}{\partial t} = -i\frac{\omega}{2}\psi$ pour l'état fondamental du piège.
6. Résoudre numériquement sur un temps $t = 1.5$ l'équation de Schrödinger pour différents états initiaux : par exemple un paquet centré en -2 initialement au repos et de largeur 0.3. On utilisera la méthode `RK23`.
7. Ecrire une fonction python `moments_x(psi)` qui retourne $\mathbb{E}(X)$ (position moyenne) et $\mathbb{E}((X - \mathbb{E}(X))^2)$ (carré de la largeur du paquet)
8. Tracer ces moments en fonction de t.
9. Question 7 et 8, mais avec l'impulsion. Dans le calcul de l'espérance, on suprimera les parties hautes impulsions qui sont dûes à du bruit numérique
10. Pendant 0.5 seconde, on laisse la paquet évoluer librement (V=0) puis pendant une durée $t < 0.5$, le paquet évolue avec le potentiel harmonique. Tracer la largeur en impulsion en fonction de cette durée.

4.3 Piège de Penning

À partir du sujet de 2021

En physique, les pièges à ions sont des dispositifs permettant de stocker des particules chargées pendant une longue durée, notamment dans le but de mesurer leurs propriétés avec précision.

Il n'est pas possible de réaliser un piège à ions uniquement avec un champ électrique statique : en effet, un tel piège nécéssiterait d'avoir un maximum ou minimum du potentiel électrique - ce qui est impossible car son Laplacien est nul.

Le piège de Penning est un dispositif permettant de stocker des particules chargées, grâce à la combinaison d'un champ magnétique uniforme et d'un champ électrique quadripolaire constant. Nous proposons d'en simuler le comportement.

On considère un ion béryllium Be^+ dont on notera q la chage. On note B l'amplitude du champ magnétique, uniforme, et dirigé selon z. Le champ electrostatique dipolaire dérive du potentiel suivant :

$$V = \frac{Q}{2}\left(x^2 + y^2 - 2z^2\right)$$

Le champ électrique vaut donc :

$$\vec{E} = -Q\left(x\vec{u}_x + y\vec{u}_y - 2z\vec{u}_z\right)$$

La force est donnée par :

$$\vec{F} = q\left(\vec{E} + \vec{v} \wedge \vec{B}\right)$$

où le symbol $\wedge$ représente le produit vectoriel.

Valeurs numériques :

```
proton_mass = 1.67E-27
m = 9*proton_mass
q = e = 1.6E-19

# Valeurs par défaut
B = 2 # Tesla
Q = -300/1E-2**2 # 300V/cm^2
```

1. Ecrire une fonction python qui renvoie l'accélération $\vec{F}/m$ en fonction de $\vec{r}$ la position et $\vec{v}$ la vitesse représentée par des tableaux numpy. On pourra utiliser la fonction `np.cross` pour calculer la produit vectoriel
2. Ecrire une fonction qui à partir d'une condition initiale r_0, v_0 renvoie la position et la vitesse pour un ensemble de valeur (croissante) t_i

```
def trajectoire(r_0, v_0, t_array, Q, B):
    ....
    return x, y, z, vx, vy, vz
```

3. Dans un champ magnétique uniforme (sans champ électrique), un particule à une trajectoire circulaire dans le plan orthogonal au champ. Vérifier ceci en calculant la tractoire d'une particule ayant initialement une vitesse de 100 m/s dans ce plan.
4. On considère maintenant une particule dans le champ magnétique et électrique. Paramètre initiaux : $\vec{r} = (0, 100\ \mu\mathrm{m}, 0)$ et $\vec{v} = (500\ \mathrm{m/s}, 100\ \mathrm{m/s}, 1\ \mathrm{m/s})$. On calculera pendant $5\ \mu$s en échantillonant sur au moins 2000 points. Tracer la trajectoire dans le plan x, y puis en 3D.

4.4 Nuage d'ions

On considère un nuage de N ions de masse m et charge q. On note $\vec{r_i}$ et $\vec{v_i}$ la position et la vitesse du ième ion. Les ions sont dans un piège électrostatique. De plus, ils interagissent entre eux par la force de Coulomb.

La force électrostatique dérive d'un potentiel électrostatique :

$$V(\vec{r}) = \frac{1}{2}k_x r_x^2 + \frac{1}{2}k_y r_y^2 + \frac{1}{2}k_z r_z^2$$

La force de Coulomb s'écrira sous la forme:

$$\vec{f_i}(\vec{r_i}, \vec{r_j}) = \kappa \frac{q^2}{|\vec{r_i} - \vec{r_j}|^3}(\vec{r_i} - \vec{r_j})$$

Pour simplifier et avoir une meilleure représentation graphique, on suprimera totalement la dimension z.

On utilisera des unités adimensionnées avec des constantes de l'ordre de 1. Par défaut on prendra : $m = 9$, $q = 1$, $k_x = 1$, $k_y = 1.3$, $\kappa = 1$.

L'état des N particules à un instant donné est représenté par 4 tableaux numpy de taille N: `r_x`, `r_y`, `v_x`, `v_y`. Toutes les fonctions seront écrites avec ces variables. On ne regroupera les 4 tableaux que dans la fonction `f(t, y)` qui sera utilisée par `solve_ivp`.

Le tableau `y` est défini en rassemblant les 4 tableaux numpy. On définit alors les fonctions `join` et `split` (voir ci-dessous)

Les paramètres seront des constantes globales.

1. Écrire la fonction `force_piege(r_x, r_y)` qui renvoie la force dérivant du potentiel (`f_x` et `f_y`).
2. Écrire la fonction `force_coulomb(r_x, r_y)` qui renvoie f_x et f_y, la force de Coulomb. On ne cherchera pas à éviter les boucles for dans cette fonction.
3. Écrire la fonction `f(t, y)` qui définit la dynamique du problème.
4. On condidère un nuage avec une distribution initiale donnée par une loi de Maxwell-Boltzmann (avec $k_B = 1$ et $T = 1$) pour les particules sans interaction. Calculer jusqu'au temps $T = 20$ l'évolution de la position des particules.
5. Vérifier que l'énergie totale est conservée. On utilisera les fonctions ci-dessous.
6. Faire une animation (voir exemple de code ci dessous)

7. On rajoute une force de dissipation, selon l'axe x :

$$\vec{F}_{i,x} = -\alpha v_{i,x}$$

   Simuler l'experience en prenant $\alpha = 0.1$ et $N = 20$. Que se passe-t-il au temps long ?

8. Utiliser le decorateur `jit` de numba pour accélérer le code. Faire une comparaison de vitesse.

9. On considère un nuage contenant deux types de particules (masse m_1 et m_2) et de même charge $q = 1$. Seuls les particules de type 1 sont soumises à la force dissipative. Simuler l'expérience en prenant les mêmes paramètres qu'à la question 7. On prendra par exemple $m_1 = 9$, $N_1 = 20$ et $m_2 = 2$ et $N_2 = 5$

```
# Paramètres
m = 9; q = 1
k_x, k_y = 1, 1.3
kappa=1
k_B = 1
T = 1
```

Quelques fonctions pour vous aider :

- Fonction pour passer de 4 tableaux à un seul et réciproquement

```
def join(r_x, r_y, v_x, v_y):
    return np.concatenate((r_x, r_y, v_x, v_y))

def split(y):
    N = len(y)//4
    return y[:N], y[N:2*N], y[2*N:3*N], y[3*N:4*N]
```

- Calcul de l'énergie

```
def energie_cinetique(r_x, r_y, v_x, v_y):
    return np.sum(.5*m*v_x**2 + .5*m*v_y**2)

def energie_piege(r_x, r_y):
    return np.sum(k_x*r_x**2/2 + k_y*r_y**2/2)

def energie_coulomb(r_x, r_y):
    N = len(r_x)
    total = 0
    for i in range(N-1):
        for j in range(i+1, N):
            d2 = (r_x[i]-r_x[j])**2 + (r_y[i]-r_y[j])**2
            total += kappa*q**2/np.sqrt(d2)
    return total

def energie_totale(r_x, r_y, v_x, v_y):
    return (energie_coulomb(r_x, r_y) +
            energie_piege(r_x, r_y) +
            energie_cinetique(r_x, r_y, v_x, v_y))
```

```
from IPython.display import HTML
from matplotlib.animation import FuncAnimation
```

(continues on next page)

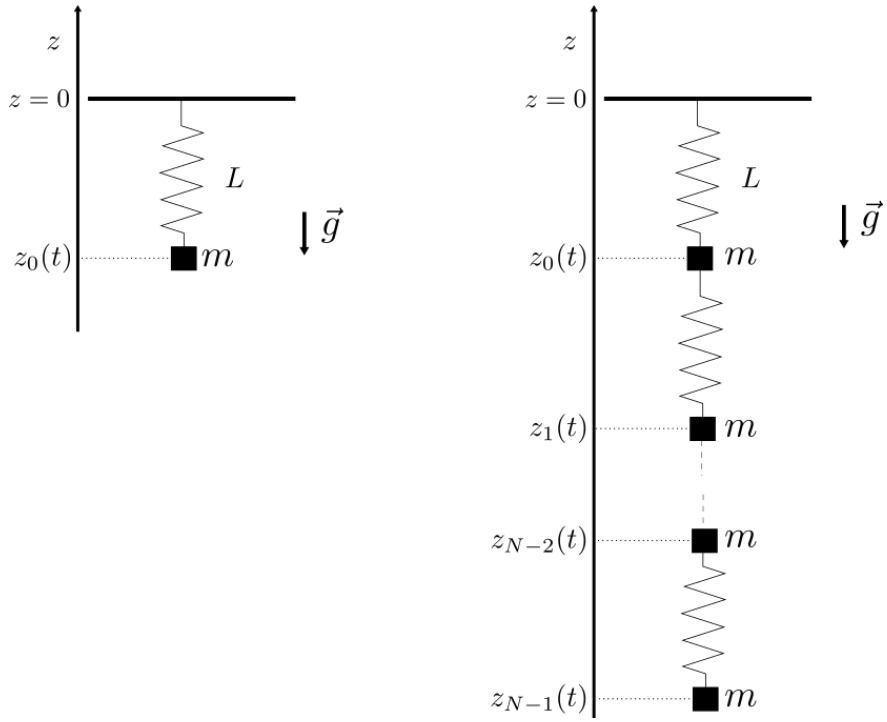
(continued from previous page)

```
fig, ax = plt.subplots()
l, = ax.plot(split(res.y[:,0])[0], split(res.y[:,0])[1], 'o')
ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)

def animate(i):
    r_x, r_y, v_x, v_y = split(res.y[:,i])
    l.set_data(r_x, r_y)

ani = FuncAnimation(fig, animate, frames=len(res.t))


HTML(ani.to_jshtml())
```

4.5 Chaîne de ressorts

Examen 2022

On considère pour commencer une masse m suspendue à un ressort de raideur k et de longueur au repos L. A l'équilibre la masse est à la position $z_{eq} = -L - a_{eq}$ où a_{eq} est l'allongement du ressort. On fera attention aux signes !



On prendra $k = 10$ N/m, $m = 0.03$ kg, $g = 9.81$ m/s^2 et $L = 30$ cm.

1. Ecrire une fonction Python qui renvoie l'accélération de la masse en fonction de sa position z et du champ de gravité g.
2. Résoudre numériquement l'équation différentielle avec pour condition initiale $z = -L$ et $v_0 = 0.1$ m/s et donner la position et la vitesse après 0.1 s.
3. Tracer la position sur environ 10 périodes.

4. Vérifier que l'énergie totale du système est conservée.
5. Ajuster votre graphique par une sinusoïde et vérifier que l'on a bien la relation $\omega = \sqrt{k/m}$.
6. Retrouver la fréquence d'oscillation cette fois en calculant la transformée de Fourier numérique de la position. Comment améliorer la résolution de la valeur déduite ?

On considère maintenant une chaîne de masses reliées par des ressorts. On prendre $N = 10$ masses numérotées de 0 à $N - 1$.

On note $z_i(t)$ la position de la masse numéro i et $l_i(t)$ la longueur de chaque ressort ($l_i(t) > 0$). On a $z_i(t) = -\sum_{j=0}^{j=i} l_j(t)$

7. Ecrire une fonction qui à partir d'un tableau des positions z_i renvoit un tableau des alongements $a_i = l_i - L$.

L'accélération d'une masse i est donnée par :

$$\frac{d^2 z_i}{dt^2} = -g + \frac{k}{m}(a_i - a_{i+1}) \tag{1}$$

étant entendu que pour la dernière masse, il n'y a pas de terme en a_{i+1}.

8. Ecrire une fonction qui renvoie sous forme d'un tableau l'accélération en fonction des positions z_i.

À l'équilibre, en présence du champ de gravité, l'alongement a_i est donné par la masse totale située en dessous ($j \geq i$) : $a_i^{\text{eq}} = (N - i)\frac{mg}{k}$.

9. Vérifier que pour cette position, l'accélération est bien nulle. (Il s'agit ici de tester la fonction écrite à la question précédente.)
10. Résoudre l'équation différentielle en prenant comme conditions initiales, pour chacune des masses, une vitesse nulle et une position déplacée de 20 cm vers le haut par rapport à sa position d'équilibre en présence du champ de gravité.
11. Qu'observez-vous dans le spectre du mouvement ?
12. On considère maintenant qu'à l'instant $t = 0$, le système est à l'équilibre. Cependant, on coupe le ressort numéro "0". Ecrire une nouvelle fonction qui renvoit l'accélération en fonction des positions.
13. Résoudre l'équation différentielle. Tracer la position de la masse numéro $N - 1$ en fonction de t jusqu'à environ 0.4 s. Qu'observe-t-on ?
14. Tracez la position des masses numéro 0, 4 et 9 dans le référentiel en chute libre. Tracer aussi dans ce référentiel la position du centre de masse (position moyenne des masses).

Fonctions utilisées

```
from scipy.optimize import curve_fit
from numpy.fft import rfft, rfftfreq
from scipy.integrate import solve_ivp
```

5 Transformée de Fourier

5.1 Exemple simple de DFT

L'objectif de cet exercice est d'étudier numériquement la transformée de Fourier d'une Gaussienne : $G(x) = \exp(-\frac{x^2}{2\sigma^2})$. Analytiquemet, la T.F. de G vaut $\tilde{G}(\omega) = \sqrt{2\pi}\sigma \exp(-\frac{\omega^2\sigma^2}{2})$

1. Écrire une fonction Python qui retourne une Gaussienne $\exp(-\frac{x^2}{2\sigma^2})$ sur [-L/2, L/2] et son axe des \texttt{x = L/N*np.arange(N) - L/2}, prenant comme paramètres σ, L et N.

2. Tracer la Gaussienne pour $\sigma = 0.5$ et $N = 1000$ et $L = 5$.
3. Tracer la DFT de la Gausienne pour $\sigma = 0.15$ pour une plage de fréquences inférieures à 20 : (a) Quel taux d'échantillonnage $\Delta x = L/N$ faut-il utiliser ? (b) Quelle valeur minimum de L faut-il choisir pour obtenir une résolution en fréquence de 0.04 ?
4. Comparer la courbe obtenue avec la formule analytique ? Pourquoi la DFT de la Gaussienne n'est pas possitive ?
5. Que se passe-t-il si $\sigma = 1.5$?
6. Appliquer la transformée de Fourier inverse et vérifier que l'on obient bien le signal initial.

5.2 Machine à laver le linge

Le fichier `data/machine_a_laver.wav` contient le bruit d'une machine à laver le linge lors de l'essorage. Quelle est la fréquence de rotation du tambour ?

5.3 Les verres musicaux

Un verre d'eau partiellement rempli peut émettre une note distincte lorsque l'on frotte son bord avec un doigt mouillé. C'est l'ingrédient de base du Verrillon. Mais que cache son spectre ?

La vidéo `data/Singing_Glass.MOV` à été faite avec un verre à vin, de l'eau et un smartphone. Sa bande son à été extraite dans le fichier `data/SingingGlass.wav`.

1. Télécharger, puis importer la piste sonore SingingGlass.wav :
2. Afficher le nombre de points N contenus dans ce signal, la durée de l'intervalle entre deux points et la durée totale de l'enregistrement.
3. Afficher le signal. Quelle est approximativement la fréquence du son ?
4. L'intensité du spectre à une fréquence donnée est proportionnel au carré du module de la transformée de Fourier. Affichez le spectre de SingingGlass.wav.
5. Zoomer autour de la fréquence du son. Que remarquez-vous ?
6. On a phénomène de battement que l'on entend à l'oreille. Qu'elle est la fréquence du battement ?

Il s'agit d'un phénomène méconnu mais relativement commun avec les verres musicaux, où deux modes de vibration orthogonaux sont excités. Ces deux modes n'ont pas exactement la même énergie due aux imperfections dans la symétrie du verre. La non-dégénérescence de ces modes dépend du niveau d'eau. Voir par exemple: https://newt.phys.unsw.edu.au/music/people/publications/Jundtetal2006.pdf

On observe bien deux fréquences séparée d'environ 2 Hz.

6 Problèmes

6.1 Analyse de spectres de l'hydrogène

Les données que nous allons analyser sont des données de spectroscopie de l'atome de deutérieum (l'isotope de l'hydrogène de masse atomique 2). Dans le modèle simple de Bohr, les niveaux d'énergies sont donnés par :

$$E_n = -hc\frac{R_D}{n^2}$$

où h et la constante de Planck, c la célérité de la lumière et R_D la constante de Rydberg du deutérium qui est reliée à la constante de Rydberg R_∞ par :

$$R_D = \frac{R_\infty}{1 + \frac{m_e}{m_D}}$$

où m_e est la masse de l'électron et m_D celle du Deuteron (Noyau de l'atome de deutérium : un proton et un neutron). La constante R_∞ correspond à la constante de Rydberg pour un noyau de masse infinie.

La transition que l'on étudie est la transition entre les niveau $n = 1$ et $n = 3$. Sa fréquence vaut :

$$\nu_{\text{Bohr}} = \frac{E_3 - E_1}{h}$$

- Constante de rydberg : $R_\infty = 10973731.56816\ \text{m}^{-1}$
- Rapport de masse : $\frac{m_D}{m_e} = 3670.4829679$
- Vitesse de la lumière dans le vide: $c = 299792458$ m/s

1. Calculer la fréquence de la transition dans le modèle de Bohr et donner sa valeur en MHz. On formatera la valeur pour avoir trois chiffres après la virgule (précision au kHz).

Le principe de la spectroscopie consiste a illuminer un jet d'atomes à l'aide d'un laser dont on connait la fréquence et à mesurer la fluorescence des atomes. Cette fluorescence étant très faible, on utilise un compteur de photon pour la mesurer.

Les données sont dans le fichier `data/data_hydrogene_simple.txt`. Il s'agit d'un fichier texte simple. Il comprend une colonne qui correspond à la fréquence (mesurée) du laser de spectroscopie et une colonne qui correspond au nombre de photons détectés sur une seconde de mesure. Ce nombre est proportionnel au nombre de photons absorbés par les atomes d'hydrogènes.

2. Lire et afficher les données.

On va ajuster le profile par une fonction lorentzienne :

$$a + \frac{b}{1 + \left(\frac{f - f_0}{\Delta f}\right)^2}$$

où a est un offset, b une amplitude, f_0 la fréquence centrale et Δf la largeur en fréquence.

3. Faire un ajustement des données par une lorentzienne (que l'on représentera graphiquement).

Les données sont prises par rapport à une fréquence de référence, c'est à dire que la fréquence du laser est la somme entre la fréquence de référence ($f_{\text{ref}} = 2923538429.3$ MHz) et la fréquence des données.

4. Donner la valeur de la fréquence absolue ν ainsi que l'incertitude absolue σ_ν et relative $\sigma\nu/\nu$) de la mesure du pic. La différence par rapport à la valeur du modèle de Bohr est dûe a des corrections supplémentaires (effet du spin de l'électron, de la taille fini du noyau, effets relativistes, ou encore de la polarization du vide qui peuvent être calculés à l'aide de l'Electrodynamique Quantique). Calculer cet écart.

5. Les données correspondent en réalité à 6 spectres de N points qui ont été enregistrés à la suite. Ajuster chacun des 6 spectres pour en déduire leurs fréquences absolues et leurs incertitudes que l'on affichera.

6. Tracer avec des barres d'erreurs les points sur un graphique ayant comme axe des abscisses le numéro du spectre (1, 2, .., 6). Représenter sur ce même graphique la valeur moyenne des fréquences $\bar{\nu}$ et son écart type σ_ν (on utilisera une ligne continue rouge pour la moyenne et des lignes discontinues noires pour $\bar{\nu} \pm \sigma_\nu$). Est-ce que l'écart type des 6 fréquences est en accord avec l'incertitude obtenue à partir de l'ajustement de l'ensemble des données (à la question 4).

Les données du fichier `data_hydrogene_simple.txt` sont en fait le centre d'un spectre plus large. Le fichier `data/data_complete.dat` contient tous les points de mesures. Ce fichier est formaté légèrement différement (il contient un en-tête et un ';' comme séparateur).

7. En utilsant les options de `np.loadtxt` lire le fichier et tracer les points.

Ce spectre peut être modélisé par la somme d'une gaussienne (fond large) et d'une lorentzienne (étroite).

8. Faire l'ajustement. Donner la valeur de la position du centre de la lorentzienne étroite et son écart type. Comparer ces résultats à ceux obtenus à la question 4.

8. A l'aide d'un masque, ajuster à l'aide d'une lorentizienne uniquement les points obtenus pour une fréquence laser comprise entre -2 et 2 MHz. Comparez avec la question 4.

Fonctions utilisées

```
from scipy.integrate import solve_ivp
from matplotlib.pyplot import errorbar
# ou ax.errorbar
```

6.2 Modèle d'Ising

Un système magnétique peut-être décrit de façon simple avec l'Hamiltonien suivant (modèle d'Heisenberg):

$$H = -\frac{4}{h^2}\sum_{i<j} J_{ij}\vec{s_i}\cdot\vec{s_j} - \gamma\vec{B}\cdot\sum_i \vec{s_i} \tag{2}$$

Dans cet exercice, nous allons nous intéresser à un modèle encore plus simple: celui d'un ensemble de spin 1/2 et ne considérer que l'intéraction entre les spins voisins les plus proches. En écrivant $\vec{s_i} = \frac{\hbar}{2}S_i$ ($S_i = \pm 1$), on obtient le modèle d'Ising décrit par l'Hamiltonien:

$$H = -J\sum_{\langle i,j\rangle} S_i S_j - \mu B\sum_i S_i \tag{3}$$

où la notation $\langle i,j\rangle$ denote la somme sur les paires des plus proches voisins.

Nous allons dans cet exercice simuler ce système à l'aide d'un algorithme appelé "Metrolopolis". Celui-ci est détaillé sur la page wikipedia dédié au model d'Ising. https://en.wikipedia.org/wiki/Ising_model . Nous prendrons un champ magnétique nul.

L'état du système sera représenté par un tableau de taille $N \times N$. Nous utiliserons des conditions périodiques (i.e. le spin à la position $(N-1, j)$ est couplé au spin à la position $(0, j)$, etc.

1. Écrire une fonction `energy(state)` qui retourne l'énergie totale du système magnétique dans l'état 'state'.

2. Écrire une fonction `delta_energy(state, i, j)` qui retourne la difference d'énergie induite par le flipping du spin de coordonnees `(i, j)`. Nous rappellons que seule l'intéraction entre plus proches voisins est considérée.

3. Écrire une fonction `metropolis(state, beta, M)` qui retourne l'état du système magnétique après avoir implémenté M fois l'algorithme de Metrolopolis. Une réalisation de l'algorithme correspond à choisir un point de manière aléatoire, à calculer la différence d'énergie induite par le flip de celui-ci. Si cette énergie est négative, alors on considère que le spin flip. Si la différence d'énergie est positive, on autorise le spin flip selon la probabilité: $e^{-\beta\Delta E}$.

4. Afficher l'image pour $\beta = 1$, après 1000, 1 000 000, 100 000 000 itérations. Vous pouvez aussi faire une animation. On prendra comme état initial, 300x300 spins de valeur aléatoire ± 1. Il faudra utiliser numba.

5. Nous allons regarder l'énergie moyenne du système. Modifier votre fonction metropolis, de façon à pouvoir enregistrer l'énergie moyenne du système pendant l'évolution. Pour éviter des problèmes de mémoires, nous vous conseillons de l'enregistrer uniquement tous les N^2 réalisations.

   Tracer l'énergie du système au cours du temps. Observer le temps de convergence du système (on prendra $\beta = 1$, $\beta = 0.2$ et $\beta = 0.1$). En plus de la convergence, qu'observez-vous pour $\beta = 0.2$?

6. Tracer l'énergie moyenne en fonction de β. Vous pourrez aussi accélérer votre calcul en l'évaluant sur plusieurs noyaux de votre ordinateur, à l'aide du décorateur: `@jit(parallel=True)` et de la boucle `for beta in numba.prange(xxx):`.
7. De même, tracer les fluctuations d'énergie en fonction β
8. Comme vous l'avez vu en cours de Physique Statistique, il existe une solution analytique au modèle d'Ising en deux dimensions:

$$Z = \left(2\cosh\frac{2J}{k_BT}e^I\right)^N, \quad I = \frac{1}{2\pi}\int_0^\pi \mathrm{d}\phi \log\frac{1+\sqrt{1-x^2\sin^2\theta}}{2}, \quad x = \frac{2\sinh\left(\frac{2J}{k_BT}\right)}{\cosh^2(\frac{2J}{k_BT})}$$

Comment calculer l'énergie moyenne du système à partir de la fonction de partition Z ? Evaluez numériquement cette expression et comparez à votre simulation metrolopolis.

Si vous avez le temps, même question pour les fluctuations d'énergie

9. La transition de phase a lieu pour T qui vérifie:

$$\sinh\left(\frac{2J}{kT}\right) = 1 \tag{4}$$

Résolvez cette equation numériquement (on pourra utiliser la fonction root de scipy) et comparez aux résultats de la question précédente.

On observe des fluctuations importantes pour la courbe $\beta = 0.2$

L'énergie moyenne est donnée par :

$$\langle E\rangle = -\frac{\partial \ln Z}{\partial\beta} \tag{5}$$

Les fluctuations d'énergie par la dérivée seconde.