Examen de TP, fin du second semestre

Ouvrez dans pyzo le fichier `TP_note_sem2.py` situé dans le répertoire `Devoirs/fleck/../sources/TP_note_`
N'oubliez pas que l'exécution doit se faire via `Ctrl-SHIFT-E` pour que tout marche bien.

Introduction

L'objectif de ce TP est d'étudier la lecture, l'affichage et le traitement automatique de spectres RMN que vous avez rencontré dans les cours de chimie. Aucune connaissance de chimie n'est nécessaire pour effectuer correctement le TP, mais cela ne m'empêchera pas de donner quelques infos intéressantes du point de vue chimique de temps en temps.

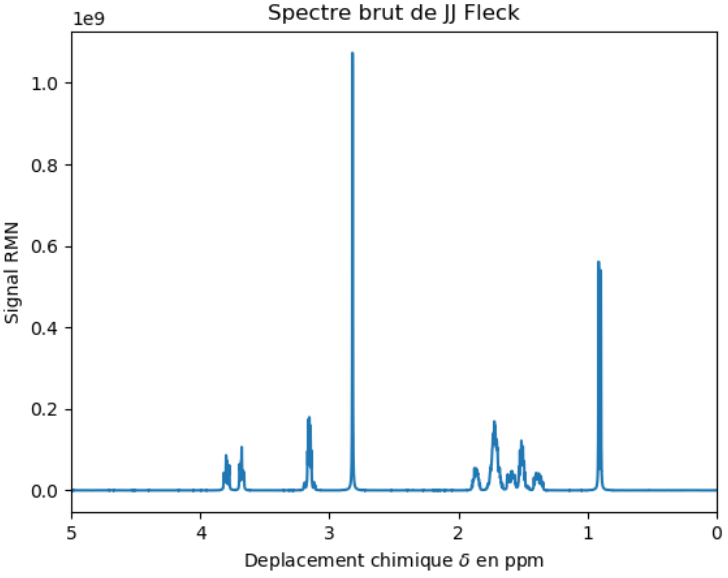
Notez que la dernière question de chaque partie s'appuie sur ce qui a été fait plus haut mais en-dehors de celles-ci chaque question peut être traitée indépendamment du reste et constitue un problème algorithmique en soi. Notez aussi que les graphiques compteront chacun pour 5 points mais leur tracé est appelé automatiquement par les tests (si vos fonctions sont bien écrites).

Partie I

Lecture/affichage du spectre RMN

Le spectre RMN que l'on va plus particulièrement étudier est stocké dans le fichier `spectrum.jdx`. Il est constitué (ici, mais cela peut changer) de 17 710 lignes de données réparties sur trois colonnes [1].

1. Compléter la fonction `lecture_fichier(filename)` qui doit lire le fichier (dont le nom est donné en argument et sans oublier de refermer le descripteur de fichier) et renvoie sous forme de listes ou de `np.array` les deux premières colonnes du fichier, la première donnant les déplacement chimique δ et la seconde le signal correspondant à ce déplacement chimique (vous pouvez gentiment ignorer la troisième colonne qui correspond au signal intégré et dont on va retrouver les informations par la suite).
2. Écrire une fonction que l'on nommera `visualise_spectre(delta,spectre)` et qui, étant donné une suite de déplacement chimique et les valeurs correspondantes du spectre, trace le spectre RMN et le sauvegarde dans un fichier `mon_spectre_brut.png`. Vous rajouterez votre nom dans le titre (par exemple « Spectre de JJ Fleck ») ainsi que les indications « Déplacement chimique » et « Signal RMN » respectivement sur les axes des abscisses et des ordonnées. Comme les chimistes dessinent historiquement les spectres RMN avec les plus grands déplacement chimique du côté gauche du graphique (et non du droit comme on s'y attendrait), on pourra utiliser la commande `plt.gca().invert_xaxis()` pour inverser l'axe des x ou alors imposer les limites « à l'envers » via `plt.xlim(5,0)`.



1. Attention à bien écrire des programmes généraux car je changerai le spectre effectivement lu pour les tests finaux, impossible donc de « hardcoder » les valeurs attendues car elles changeront lors de l'évaluation finale...

Partie II

Intégration du signal spectroscopique

Les massifs de pics que l'on observe sur le spectre précédent sont directement reliés à l'environnement des hydrogènes dans la molécule étudiée. En revanche, le nombre d'hydrogènes équivalents concernés est relié à l'aire sous les pics donc à l'intégrale du spectre lu précédemment.

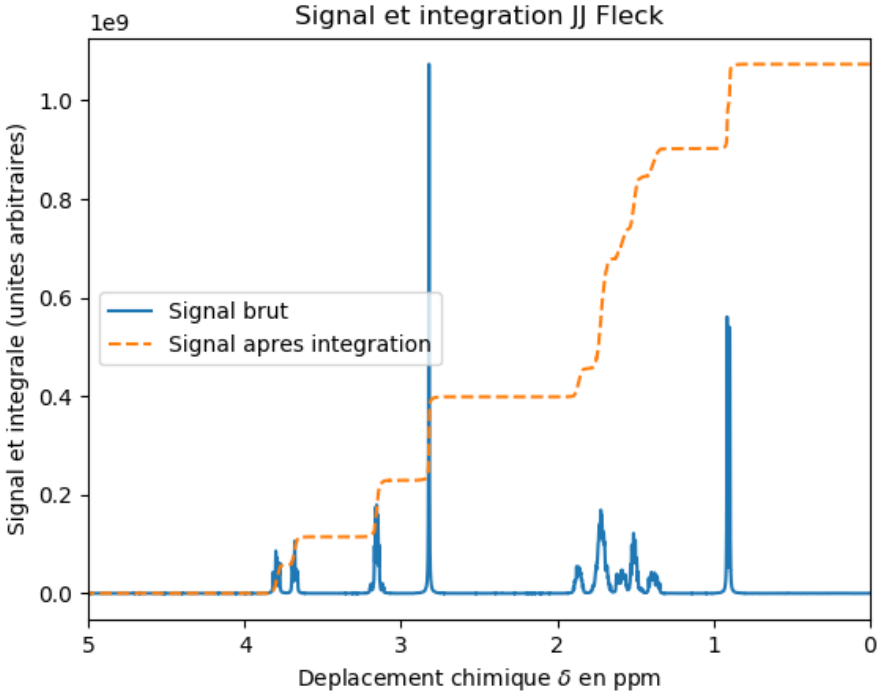
3. Écrire une fonction `integration_trapeze_cumulee(`x`,`f_x`)` qui, pour un ensemble d'abscisses x associées aux valeurs de la liste f_x, renvoie la liste des intégrales cumulées jusqu'à $x[i]$. En termes mathématiques, on veut la liste `integrale` telle que

$$\forall i \in [\![0 \,;\, \texttt{len}(x) - 1]\!] \qquad \texttt{integrale[}i\texttt{]} = \int_{x[0]}^{x[i]} f_x(t)\,\mathrm{d}t$$

où l'intégrale doit être calculée par la méthode des trapèzes.

NB : il est *interdit* d'utiliser `np.trapz` ou son cousin `cumtrapz`.

4. Écrire une fonction `visualise_spectre_integre(delta,spectre)` qui trace à la fois le spectre et son intégrale [2] en fonction du déplacement chimique δ (à nouveau, on utilisera la commande `plt.gca().invert_xaxis()` pour inverser l'axe des abscisses ou alors on peut utiliser `plt.xlim(5,0)` pour obliger les déplacements chimiques à aller de 5ppm à 0ppm dans cet ordre) et sauvegarde le résultat dans le fichier [3] `mon_spectre_integre.png` en précisant votre nom dans le titre, ce qu'il faut sur les axes. Attention, du fait de l'ordre (décroissant) dans lequel sont donnés les déplacements chimique, l'intégrale est négative. On tracera donc sa valeur absolue et on se débrouillera pour que la valeur maximale du signal intégré corresponde à la valeur maximale du spectre [4] comme sur l'exemple suivant.



2. À vous de la calculer au vol en utilisant la fonction précédente.
3. Petit rappel : Pas d'accents, pas d'espaces !
4. Il peut être intéressant de convertir toutes ces choses en `np.array` pour aisément les transformer.

Partie III

Détection des plateaux

Pour trouver la position des plateaux qui permettront de calculer le nombre d'hydrogènes équivalents présents dans chaque saut, plusieurs choix s'offrent à nous. On va ici essayer de caractériser un plateau comme une zone où la dérivée est « presque nulle ». Comme on regarde le signal intégré, cela revient à dire que le signal du spectre initial est « presque nul ». On dira pour simplifier que « presque nul » signifie que la valeur du signal est inférieure à 0.5% du pic principal du signal.

5. Écrire une fonction `rabotage(signal)` qui renvoie une liste (ou un `np.array`) dont les valeurs « presque nulles » (au sens défini au-dessus) de la liste `signal` sont mises effectivement à 0, les autres étant laissées inchangées.

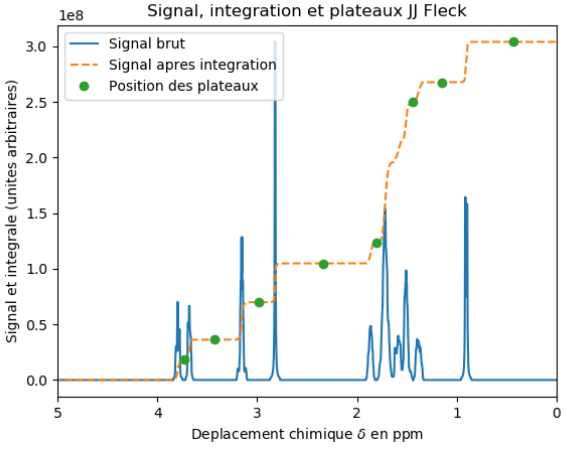
Pour éviter les « glitchs » dû à l'effet de seuil précédent, on veut appliquer un filtre passe-bas qui écarte les fluctuations à hautes fréquences [5]. L'idée est de regarder un extrait de demi-largeur `etendue` autour du point `i` considéré et remplacer la valeur du signal par la médiane calculée sur cet extrait. On rappelle que l'on peut facilement extraire une sous-liste d'une liste (ou d'un `np.array`) avec la notation `extrait=liste[i-etendue:i+etendue+1]` et que numpy permet de prendre facilement la médiane par `mediane = np.median(liste)`.

6. Écrire une fonction `filtre_median(x,fx,etendue)` qui applique le procédé précédent sur le signal contenu dans la liste [6] `fx`. La fonction doit renvoyer deux listes (ou deux `np.array`), la première est la liste des `x` amputée de ce qu'il faut pour correspondre à la deuxième qui contient l'application du filtre médian à `fx`.
   À titre d'exemple, voici ce qu'on devrait obtenir avec une étendue de 1 (donc des groupements de trois valeurs consécutives) sur une petite liste

```
>>> x = [1 , 2,  3, 4, 5,  6, 7, 8,   9]
>>> fx= [11,12,100,13,14,254,15,16,-124]
>>> filtre_median(x,fx,1)
[2,3,4,5,6,7,8], [12,13,14,14,15,16,15]
```

Maintenant que le bruit a été filtré et raboté, on peut supposer que notre signal contient de longues plages de 0 ponctuées par des zones non nulles correspondant aux pics du signal RMN.

7. Écrire une fonction `detection_plateaux(signal_propre)` qui prend en argument le signal (proprifié par les méthodes précédentes) et qui doit renvoyer la liste des positions du milieu (donc données par `(debut+fin)//2` pour avoir des entiers) de chaque intervalle où le signal est (strictement) nul.
8. Écrire finalement une fonction que l'on nommera `visualise_plateaux(delta,signal)` et qui, à partir du signal brut (non proprifié) utilise toutes les fonctions précédentes pour stocker dans le fichier `mon_spectre_plateaux.png` la représentation
   - du spectre brut ;
   - du spectre intégré (où le maximum est ajusté au maximum du spectre brut) ;
   - des positions des plateaux sous forme de points ronds comme dans l'exemple ci-contre.



5. C'est-à-dire les zones où le signal s'écarte rapidement de sa valeur normales avant d'y revenir.
6. `x` et `fx` peuvent aussi être des `np.array`, votre procédure doit fonctionner correctement quel que soit le cas.

Partie IV

Nombre d'hydrogènes présents dans la molécule

Il ne reste plus qu'à retrouver le nombre d'hydrogène total de la molécule à partir de la connaissances des valeurs des plateaux. En effet, la hauteur des marches entre deux plateaux consécutifs est proportionnelle aux hydrogènes responsables de chaque pic. En regardant si la plus petite marche correspond à 1, 2 ou 3 hydrogènes et en vérifiant que toutes les autres marches donnent des valeurs presque entières sous cette hypothèse, on peut remonter au nombre total d'hydrogènes correspondant.

9. Écrire une fonction `trouve_pourcentages(valeurs_plateaux)` qui renvoie la liste des pourcentages de la hauteur de chaque marche comparée à la hauteur totale. Par exemple si la liste `valeurs_plateaux` vaut `[0,2.5,5,10]`, c'est qu'il y a trois marches, deux de l'ordre de 25% de la hauteur totale et la dernière de taille 50%, on doit donc renvoyer la liste `[0.25,0.25,0.5]`
10. Écrire une fonction `nombre_total_hydrogenes(pourcentages)` qui, à partir de la liste des pourcentages (obtenue par exemple à partir de la fonction précédente), renvoie la plus petite valeur du nombre total d'hydrogènes présent dans la molécule. Dans l'exemple précédent, ce serait 4 puisque les deux premières marches peuvent chacune correspondre à un hydrogène alors que la dernière, deux fois plus grande, correspond alors à deux hydrogènes.
11. Écrire finalement une fonction complète `analyse_molecule(delta,spectre)` qui, à partir de la donnée du spectre brut, renvoie le nombre total d'hydrogènes de la molécule en utilisant l'ensemble des fonctions précédentes [7].

7. Une étendue de 30 points devrait être adéquate pour le filtrage.