Peak fitting: une guideline (basée sur Deep picker)

Sources:

- Les codes sont disponibles sur le dépôt github de deep picker **main_peak_fitting.cpp, spectrum_fit.h/cpp**
- L'article de DeepPicker
- La documentation du dépôt GitHub
- Des principes un peu généraux de fitting de pics RMN 2D (Gaussian, Voigt, Voigt-Lorentz)

Objectif du fitting

- Le fitting consiste à ajuster une fonction mathématique sur des données expérimentales issues d'analyse par RMN afin d'extraire des paramètres précis décrivant les pics d'un spectre 2D. Cela à pour but de quantifier les amplitudes, localiser précisément les pics, estimer les largeurs et reconstruire un spectre analytique pour permettre l'analyse de ces pics.
- Ce que j'ai moi : Avant : on a obtenu les positions approximatives des pics grâce au CNN ⇒ on a une variable **peaks_clean_filtered** (contient ppm_x, ppm_y et intensity relative car normalisée) qui se présente comme un data frame :

```
> str(peaks_clean_filtered)

'data.frame':	592 obs. of  6 variables:

 $ F1        : num  2 2 2 3 3 3 4 4 4 5 ...

 $ F2        : num  456 455 457 455 457 456 456 455 457 456 ...

 $ Intensity : num  0.15 0.155 0.144 0.154 0.144 ...

$ cluster_db: int  1 1 1 1 1 1 1 1 1 1 ...

 $ F1_ppm    : num  8.42 8.42 8.42 8.4 8.4 ...

 $ F2_ppm    : num  4.38 4.39 4.37 4.39 4.37 ...
```

Les étapes principales sont les suivantes :

1. **Détection initiale des pics**
    - Identifier les maxima dans le spectre.

- OU par méthode des réseaux de neurones convolutifs.(dans notre cas à nous)

2. **Initialisation des paramètres**

   - Définir la forme de chaque pic : gaussienne, Lorentzienne ou Voigt (=mélange des deux).

   - Estimer les paramètres initiaux : amplitude, largeur approximative, coordonnées (x, y).

3. **Définition des zones de fitting**

   - Pour chaque pic, définir une fenêtre autour de la position initiale et ainsi déterminer quelles données seront utilisées pour le fitting.

   - Identifier les pics trop proches et les regrouper en clusters pour pouvoir faire un fitting simultané, et donc éviter les interférences.

4. **Optimisation des paramètres**

   - Ajuster les paramètres du pic (position, amplitude, largeur, forme) pour minimiser l'erreur entre le spectre expérimental et le modèle analytique que l'on a réalisé.
     ⇒ Méthodes utilisées : descente de gradient, Levenberg-Marquardt (algo de minimisation non linéaire) ou solveurs avancés comme Ceres.

5. **Vérification et nettoyage**

   - Supprimer les pics trop proches d'un voisin plus grand ou trop faibles selon des seuils définis.

   - Contrôler la convergence de l'optimisation (nombre d'itérations et changement minimal des paramètres).

6. **Reconstruction du spectre entier avec toutes nos données analytique**

   - Combiner tous les pics fittés pour créer un spectre analytique complet.

   - Calculer le résidu, c'est-à-dire la différence entre le spectre expérimental et le modèle reconstruit, pour évaluer la qualité du fitting.

7. **Extraction finale des paramètres**

   - Récupérer pour chaque pic ses coordonnées exactes, son amplitude, ses largeurs et la forme optimisée.

    - Ces paramètres permettent de quantifier les composés, à compare les spectres entre eux et à analyser chimiquement/physiquement les composés

Classes principales et rôles (spectrum_fit, gaussian fit)

shared_data_2d

Cette classe contient des variables partagées par toutes les instances de fitting :

- **n_verbose** : niveau de verbosité pour les logs. (afficher explicitement ou non ce qui est fait en interne par les programmes)
- **error_scale** : échelle utilisée pour l'estimation de l'erreur (par Monte-Carlo).

fit_type

Cet enum définit la forme du pic que l'on souhaite fitter :

- **gaussian_type** : pic gaussien.
- **voigt_type** : pic Voigt (combinaison d'un pic gaussien et lorentzien).
- **exact_type** : fitting exact, utilisant directement les valeurs expérimentales.
- **voigt_lorentz_type** : Voigt sur l'axe x et Lorentz sur l'axe y.
- **null_type** : non défini.

gaussian_fit

Chaque objet **gaussian_fit** correspond à un pic OU un petit cluster de pics à fitter.
ATTENTION: c'est pas parce que ça s'appelle gaussian_fit que c'est uniquement gaussien, la classe contient également du fitting lorentzien ou voigt.

Variables principales

- **x, y** : coordonnées du pic.
- **sigmax, sigmay** : largeur gaussienne.
- **gammax, gammay** : largeur lorentzienne si pic Voigt.
- **a** : amplitude du pic.
- **err** : erreur du fitting pour ce pic.
- **peaks_fitting_data** : données locales autour du pic pour le fitting.

- **gaussian_fit_wx, gaussian_fit_wy** : taille maximale de la zone autour du pic.

- **too_near_cutoff**, **removal_cutoff** : critères pour éliminer des pics trop proches ou trop faibles.

- **peak_shape** : type de pic (gaussien, Voigt, Voigt-Lorentz).

Fonctions principales et adaptation à la forme du pic

- **one_fit_gaussian()** : ajuste un pic gaussien et optimise **x, y, a, sigmax, sigmay**.

- **one_fit_voigt()** : ajuste un pic Voigt, en optimisant **x, y, a, sigmax, sigmay, gammax, gammay**. La fonction utilise une convolution 2D Voigt pour calculer le spectre analytique correspondant.

- **one_fit_voigt_lorentz()** : ajuste un pic Voigt sur x et Lorentz sur y, avec optimisation de tous les paramètres nécessaires.

- **multiple_fit_gaussian()** et **multiple_fit_voigt()** : permettent de fitter plusieurs pics simultanément (si des pics se chevauchent par exemple).

- ***_convolution()** : génère le spectre analytique (càd le spectre “mathématique”) du pic à partir des paramètres optimisés pour comparaison avec le spectre expérimental.

- **run_multi_peaks()** : organise le fitting de tous les pics ou clusters.

- **generate_theoretical_spectra()** : reconstruit le spectre analytique complet à partir des pics fittés.

gaussian_fit est donc capable de s’adapter automatiquement au type de pic et à sa forme, que ce soit gaussien, Voigt ou Voigt-Lorentz.

spectrum_fit

Cette classe gère l’ensemble d’un spectre 2D et coordonne le fitting global.

Variables principales

- **spects** : les spectres originaux (matrices 2D).

- **fits** : vecteur de **gaussian_fit**, un objet par pic.

- **group** : groupes de pics proches ou chevauchants.

- **err, nround** : erreurs et nombre d’itérations pour chaque pic.

- **wx, wy** : largeur maximale utilisée pour définir les zones de fitting.
- **peak_cannot_move_flag** : indique si un pic peut être déplacé pendant le fitting.
- **excluded_peaks** : pics exclus du fitting.

Méthodes principales

- **init_all_spectra()** : charge tous les spectres à fitter.
- **peak_partition()** : regroupe les pics proches pour fitter simultanément les clusters.
- **peak_fitting()** : boucle sur tous les pics et appelle **gaussian_fit.run()** pour chacun.
- **fit_gather()** : rassemble les résultats des objets **gaussian_fit**, calcule l'erreur RMSD globale et reconstruit le spectre analytique.
- **generate_recon_and_diff_spectrum()** : crée le spectre reconstruit et le spectre résiduel (différence avec le spectre original).

En résumé, **spectrum_fit** est un peu comme le chef d'orchestre du fitting global, tandis que **gaussian_fit** est une sorte de "sous-classe" ajuste individuellement chaque pic selon sa forme.

Workflow global

1. Chargement des spectres (**init_all_spectra**).
2. Détection initiale des pics (**peak_reading_***). nous on le fait avec nos méthodes de max ou de cnn
3. Partition en clusters (**peak_partition**).
4. Fitting des pics ou clusters (**peak_fitting**).
    - **gaussian_fit.run()** décide du type de pic (gaussien, Voigt ou Voigt-Lorentz) et ajuste les paramètres correspondants.
5. Rassemblement des résultats (**fit_gather**).
6. Estimation d'erreur éventuelle (**run_with_error_estimation**).
7. Reconstruction du spectre analytique complet.

main_peak_fitting.cpp

- **Stocker et gérer les spectres bruts**
    - Hérite de la classe fid_2d et contient toutes les matrices 2D des spectres expérimentaux ou simulés (**spects**).
- **Gérer tous les pics à fitter**
    - Chaque pic est représenté par un objet de type gaussian_fit.
    - spectrum_fit possède un vecteur fits contenant tous les objets gaussian_fit correspondant aux pics détectés ou fournis (dans notre cas).
- **Organiser et gérer le fitting**
    - Initialise les paramètres globaux de fitting (largeur maximale des pics, seuils de suppression, nombre de rounds, type de pic).
    - Partitionne les pics en clusters si certains se chevauchent (**peak_partition()**).
    - Appelle le fitting de chaque pic via les méthodes de gaussian_fit (**peak_fitting()**).
- **Rassembler les résultats et reconstruire le spectre en version analytique**
    - Récupère toutes les amplitudes, positions, largeurs et erreurs une fois que le fitting a été réalisé.
    - Reconstruit le spectre analytique à partir des pics fittés et calcule le résidu par rapport au spectre original (**generate_recon_and_diff_spectrum()**).

3. **Résumé du rôle**

- spectrum_fit gère un spectre complet dans son ensemble.
- gaussian_fit ajuste un pic ou un petit cluster de pics à la fois. (en gros, gaussian fit est une sous classe utilisée par spectrum fit dans son ensemble)

4. **Workflow simplifié**

- Acquisition du spectre brut (fid_2d)
- Objet spectrum_fit
    - Partition des pics en clusters

- Appel du fitting pic par pic via gaussian_fit (ajustement de l'amplitude, de la position exacte et de la largeur à mi hauteur)

- Rassemblement des résultats et reconstruction du spectre

5. **Dans main_peak_fitting.cpp**

- L'objet spectrum_fit est créé pour lire les spectres à traiter, détecter les pics initiaux (inutile pour nous car on a déjà réalisé le peak picking), lancer le fitting des pics et reconstruire le spectres sur ces nouveaux paramètres.