TP 2 : Inférence bayésienne - MCMC

L'objectif de ce TP est de mettre en œuvre les techniques du maximum de vraisemblance et du Monte Carlo à Chaînes de Markov (MCMC) afin d'ajuster un modèle à un ensemble de données. Nous utiliserons tout d'abord des librairies classiques comme `numpy` et `scipy` puis nous utiliserons une librairie dédiée au MCMC : `emcee`. Cette mise en pratique vous permettra d'identifier les avantages et les limites du maximum de vraisemblance et du MCMC.

1 Application à des données simulées

Dans un premier temps, vous allez traiter des données simulées. On se place dans le cadre d'une analyse de la densité de matière au sein d'un amas de galaxies en fonction du rayon. On suppose que cet amas a une symétrie sphérique. On peut donc caractériser sa densité par un simple profil : $\rho(r)$. Le modèle utilisé pour simuler ce profil est celui de Navarro–Frenk–White généralisé :

$$\rho(r) = \frac{\rho_0}{\left(\frac{r}{r_p}\right)^c \left(1 + \left(\frac{r}{r_p}\right)^a\right)^{\frac{b-c}{a}}} \tag{1}$$

Vous fixerez les paramètres de pente a, b, c à 1.1, 5.5 et 0.31 par la suite. L'amplitude ρ_0 et le rayon caractéristique r_p seront des paramètres libres du modèle.
Par ailleurs, on se place dans un cas où une autre structure est en train de fusionner avec le halo principal. On va modéliser cette structure éloignée du centre de l'amas par une gaussienne d'amplitude A, de position moyenne μ et de déviation standard σ inconnues. Le profil mesuré est donc le suivant :

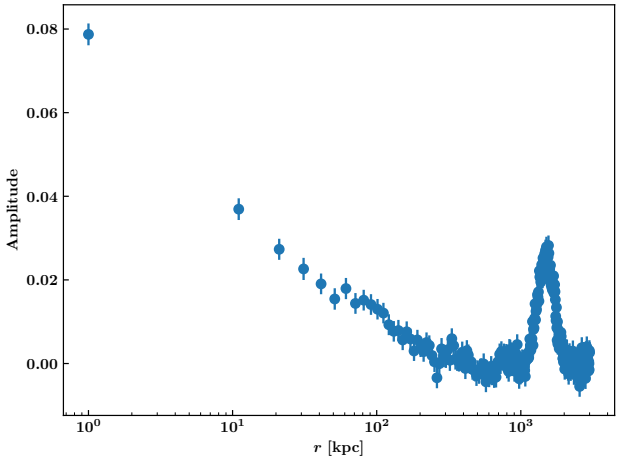

Figure 1: Profil de densité de l'amas de galaxies étudié.

Vous trouverez un fichier de données dans le dossier TP sur Moodle. C'est un fichier au format npz (ensemble de tableaux `numpy`). Ce fichier contient plusieurs entrées. Le tableau des rayons par rapport au centre de l'amas 'r', le tableau des densités mesurées 'y', le tableau des fréquences associées à la PSD du bruit de mesure 'f' et le tableau contenant la PSD du bruit de mesure 'psd'. La déviation standard du bruit associé à cette PSD est de 10^{-3}.

1.1 Minimisation du χ^2

Le premier objectif est d'estimer les valeurs de ρ_0, r_p, A, μ et σ associées au modèle ajustant le mieux les données simulées via la technique du maximum de vraisemblance. Comme notre bruit de mesure est un bruit gaussien coloré, notre fonction de vraisemblance est une loi normale et donc ce problème est équivalent à une minimisation de χ^2.

- Il est tout d'abord nécessaire d'estimer la matrice de covariance du bruit pour être en mesure de calculer la valeur du χ^2 par la suite. Définissez une fonction qui génère un tableau de bruit coloré à partir de la PSD disponible.

- Définissez une autre fonction qui calcule la matrice de covariance du bruit en exploitant la fonction précédente. En déduire la matrice de covariance inverse en utilisant la fonction `numpy` adaptée.

- Définissez une fonction qui génère un modèle pour le profil de densité de l'amas observé pour des paramètres ρ_0, r_p, A, μ et σ donnés en entrée. Le premier paramètre de cette fonction devra être le tableau des rayons par rapport au centre de l'amas.

- Compte tenu de la figure représentant les données mesurées, définissez un tableau des valeurs de départ des paramètres ρ_0, r_p, A, μ et σ qui vous semblent adaptées pour initialiser la minimisation du χ^2.

- Utilisez la fonction `curve_fit` de la librairie `scipy.optimize` afin d'estimer les meilleures valeurs des paramètres ρ_0, r_p, A, μ et σ ainsi que leur matrice de covariance associée.

- Utilisez la librairie `matplotlib.pyplot` afin de représenter les points de données ainsi que le modèle associé au maximum de vraisemblance.

- Calculez la valeur du χ^2 associé à votre meilleur ajustement. Cette valeur est-elle compatible avec la distribution de χ^2 attendue compte tenu du nombre de degrés de liberté du problème ?

- Utilisez les résultats obtenus avec `curve_fit` afin d'effectuer un tirage aléatoire de 1000 points suivant une loi normale multivariée centrée sur les meilleures valeurs de ρ_0, r_p, A, μ et σ et de matrice de covariance égale à celle renvoyée par `curve_fit`.

- Utilisez les fonctions `MCSamples` et `plots` de la librairie `getdist` afin de représenter les contours de confiance associés à votre ajustement de ces 5 paramètres.

1.2 MCMC - algorithme de Metropolis-Hastings

Vous allez maintenant mettre en œuvre une analyse MCMC afin d'ajuster le modèle de la section précédente à vos données par inférence bayésienne. Pour ce faire, vous allez développer l'algorithme de Metropolis-Hastings en n'utilisant que des fonctions de la librairie standard `numpy`.

- Définissez une fonction qui renvoie un tableau de paramètres $\vec{\theta'}$ obtenu par tirage aléatoire gaussien centré sur le tableau de paramètres $\vec{\theta_i}$ avec des déviations standards associées aux paramètres $\delta\vec{\theta}$. Cette fonction correspondra à votre fonction de proposition $g(\vec{\theta'}|\vec{\theta_i})$.

- Définissez une fonction *log_prior* qui renvoie 0 lorsque les paramètres vérifient certaines conditions et $-\infty$ (-np.inf) sinon. Cette fonction correspondra à une série de prior uniformes associés à vos paramètres.

- Définissez une fonction *log_likelihood* qui renvoie le logarithme de la fonction de vraisemblance ($-\frac{1}{2}\chi^2$) pour des paramètres donnés.

- Compte tenu de la définition du rapport d'acceptance α vue en cours, définissez une fonction *log_acceptance* qui renvoie le logarithme du rapport d'acceptance à partir des paramètres $\vec{\theta'}$ et $\vec{\theta_i}$.

- Définissez une fonction prenant $\log(\alpha)$ en paramètre et qui renvoie soit $\vec{\theta'}$ soit $\vec{\theta_i}$ en fonction du test effectué sur le rapport d'acceptance.
- Finalement, implémentez une fonction qui exploite les fonctions précédemment définies afin d'appliquer l'algorithme de Metropolis-Hastings pour un nombre de pas N donné en paramètre. Cette fonction devra renvoyer la liste des vecteurs de paramètres $[\vec{\theta_i}]_{i=1...N}$ acceptés à chaque étape.
- Initialisez votre chaîne de Markov avec des paramètres compatibles avec vos prior, définissez la taille des pas $\delta\vec{\theta}$ à considérer pour la fonction de proposition et exécutez votre code pour 10 000 pas.
- Tracez les variations de ρ_0 en fonction du pas du MCMC avec `matplotlib.pyplot`. Tracez également les variations de ρ_0 en fonction de celles de r_p pour observer l'évolution de la chaîne de Markov dans l'espace des paramètres.
- (Bonus) Généralisez votre code pour qu'il effectue l'échantillonnage des paramètres avec plusieurs chaînes de Markov.

1.3 Utilisation de la librairie `emcee` et test de convergence

Vous allez maintenant effectuer la même analyse avec une librairie python dédiée à l'échantillonnage MCMC : `emcee`. Cette librairie est basée sur l'algorithme de Metropolis-Hastings. Le code d'échantillonnage est cependant optimisé pour que la convergence des chaînes soit plus rapide. En particulier, le pas de la fonction de proposition n'est pas fixe mais est actualisé en fonction de la taille effective de la région explorée dans l'espace des paramètres.

- Définissez une fonction *log_probability* qui renvoie le logarithme de la distribution postérieure pour des paramètres $\vec{\theta_i}$ donnés. Cette fonction sera basée sur les fonctions *log_prior* et *log_likelihood* définies précédemment.
- Explorer la documentation de `emcee` afin de définir un objet *sampler* pour 10 chaînes de Markov échantillonnant la fonction *log_probability*. Lancer le MCMC pour un total de 1000 pas.
- Définissez une fonction qui réalise le test de convergence de Gelman-Rubin. Effectuez ce test de convergence en utilisant les chaînes de Markov obtenues avec `emcee` pour 1000 pas et en considérant un burn-in de 10% du nombre total de pas. On considérera que les chaînes ont convergé si $R < 1.03$ pour tous les paramètres. Si ce n'est pas le cas, combien de pas faut-il réaliser pour que la convergence des chaînes soit assurée ?
- Utilisez la fonction `function_1d` de la librairie `emcee.autocorr` afin de calculer la fonction d'autocorrélation de chaque chaîne pour chaque paramètre. Tracez ces fonctions en fonction du pas du MCMC (utilisez une échelle logarithmique pour l'axe des abscisses). À partir de combien de pas peut-on considérer que deux points des chaînes de Markov ne sont plus corrélés ?
- Utilisez la méthode `get_chain` associée à votre objet *sampler* pour définir les échantillons finaux à conserver en retirant le burn-in et en ne conservant que des points séparés du temps d'autocorrélation précédemment défini. On utilisera l'option flat = True pour combiner les 10 chaînes pour chaque paramètre.
- Utilisez les fonctions `MCSamples` et `plots` de la librairie `getdist` afin de représenter les contours de confiance associés aux échantillons conservés. Comparez ces contours avec ceux obtenus par la technique du maximum de vraisemblance en traçant les résultats de ces deux techniques sur une même figure. Interprétez les différences observées.

- Utilisez la librairie `time` afin de mesurer le temps d'exécution de votre MCMC et comparez le au temps d'exécution de la librairie `emcee`. Si vous n'avez pas généralisé votre code pour plusieurs chaînes, multipliez le temps d'exécution pour une seule chaîne par 10 pour avoir une approximation du temps d'exécution pour 10 chaînes.

- Utilisez la fonction `EmissionsTracker` de la librairie `codecarbon` afin d'estimer l'émission de CO_2 induite par l'usage de votre programme et comparez la à celle obtenue en utilisant la librairie `emcee`. Vous effectuerez un MCMC de 10000 itérations. Multipliez le résultat obtenu par votre code par 10 si vous ne considérez qu'une seule chaîne.

2 Application à un spectre réel de galaxie

Vous trouverez dans le dossier TP sur Moodle un fichier nommé f1245_888.dat. Ce fichier contient les données spectroscopiques obtenues par le KPNO (Kitt Peak National Observatory) International Spectroscopic Survey pour une galaxie appelée KISSR 127. L'objectif de cette section est d'estimer le redshift de cette galaxie en ajustant le doublet de l'atome d'oxygène doublement ionisé [OIII]. Dans le référentiel du laboratoire, ce doublet est centré sur des longueurs d'onde de 4959 Å et 5007 Å et le rapport d'amplitude entre les raies du doublet est de 1:3.

- Utilisez la librairie `pandas` pour ouvrir ce fichier de données (le séparateur entre les colonnes est une virgule).

- Utilisez la librairie `matplotlib.pyplot` afin de représenter ce spectre et identifiez à l'œil où se situe le doublet compte tenu de ses propriétés dans le référentiel du laboratoire.

- Sélectionnez la portion du spectre où la longueur d'onde est supérieure à 6600 Å afin d'étudier les propriétés du bruit de mesure. Pour ce faire, effectuez un ajustement linéaire de cette portion des données via la technique du maximum de vraisemblance (`curve_fit`). Effectuez la soustraction du meilleur modèle obtenu à cette portion des données pour que le bruit soit bien centré sur 0. Vous pouvez alors calculer la déviation standard de ce résidu. On supposera que le bruit est blanc et gaussien par la suite.

- Sélectionnez la portion du spectre où la longueur d'onde est supérieure à 6350 Å. C'est cette portion que vous allez ajuster par la suite pour obtenir le redshift de cette galaxie.

- Proposez un modèle permettant d'ajuster les deux raies du doublet et le continuum en n'utilisant qu'un seul paramètre d'amplitude pour les deux gaussiennes, un seul paramètre de largeur pour les deux gaussiennes et en utilisant le redshift ainsi que les longueurs d'onde centrales mesurées dans le laboratoire pour paramétrer la moyenne des deux gaussiennes. N'oubliez pas d'ajouter également une fonction linéaire dans ce modèle pour ajuster le continuum simultanément.

- Utilisez la librairie `emcee` pour ajuster ce modèle à la portion étudiée des données. Effectuez un premier MCMC de 5000 pas où les chaînes sont initialisées dans des prior raisonnables (utilisez la figure de votre spectre pour définir les bornes de vos prior). Repérez le pas du MCMC où le χ^2 ($-2 \times$ log_prob) est minimum. Utilisez les valeurs des paramètres correspondant à ce minimum pour initialiser les chaînes d'un deuxième MCMC de 5000 pas. Effectuez une étude de convergence et d'autocorrélation des chaînes pour sélectionner les échantillons finaux. En déduire le redshift de cette galaxie.