1 Développement des mocks

Dans ce chapitre, nous présentons la construction des *mocks* : des spectres de quasars simulés, dont les forêts Lyα sont corrélées entre elles et avec le champ de quasars sous-jacent. Ces mocks visent à reproduire les données d'eBOSS et de DESI. Ils sont nommés `SaclayMocks` et présentés dans **prov** Le code est écrit en Python [1] et se trouve en accès libre sur GitHub [2]. L'utilisation de ces mocks et leur validation seront présentés dans les chapitres suivants.

1 Objectifs des mocks

Contrairement à ce qu'on appelle les simulations, les mocks ne contiennent pas de physique à proprement parler : ils ne sont pas utilisés afin de déduire des paramètres astrophysiques. Certaines simulations, les simulations hydrodynamiques, permettent de mesurer des effets astrophysiques, comme par exemple le biais de l'hydrogène ou du Lyα. Mais ces simulations sont très couteuses car elles nécessitent de modéliser les effets physiques qui affectent les paramètres mesurés. Les mocks, quant à eux, sont conçus afin de répliquer rapidement un jeu de données, dans le but de tester l'analyse qui sera appliquée sur ces données. Les mocks sont donc utilisés afin

- de vérifier la mesure des paramètres $\alpha_\parallel$, $\alpha_\perp$: cette mesure est-elle non biaisée ?
- d'identifier les potentielles systématiques : la présence de métaux et d'HCD dans les données est-elle bien modélisée ? Affecte-t-elle la mesure de $\alpha_\parallel$, $\alpha_\perp$?
- de tester la matrice de distorsion : la distorsion de la fonction de corrélation due à l'ajustement du continuum du quasar est-elle correctement prise en compte par la matrice de distorsion ?
- de vérifier l'estimation de la matrice de covariance : la matrice de covariance, calculée à partir des données, est-elle bien estimée ?

La production et l'analyse d'un grand nombre de mocks permet de répondre précisément à ces questions. Ces mocks sont donc nécessaires pour pouvoir valider l'analyse menée sur les données.

Les mocks décrits dans ce manuscrit s'inscrivent dans les projets eBOSS et DESI. Ils sont utilisés dans l'analyse Lyα des données complète d'eBOSS (**prov**), et seront utilisés dans l'analyse Lyα de DESI. L'objectif de ces mocks est donc de répliquer au mieux les données Lyα d'eBOSS et de DESI. Ces relevés couvrent un volume de plusieurs dizaines de Gpc^3, et les échelles sondées grâce au Lyα descendent jusqu'à la centaine de kpc. Les mocks nécessitent donc de reproduire un volume immense, avec une bonne résolution. Les simulations dites *N-corps* sont des simulations qui traitent le problème à N corps. Elles sont initialisées à un redshift élevé ($z > 100$), avec une distribution de matière noire représentée par des macro-particules. Actuellement, les simulations les plus performantes utilisent environ $(10\,000)^3$ macro-particules. La masse de ces particules dépend du volume simulé. Par exemple, la simulation Outer Rim (Heitmann et al. 2019) simule un volume d'environ $(4\,\mathrm{Gpc})^3$, et les particules possèdent une masse $\sim 10^9 M_\odot$. Puis, à chaque pas de temps, ces macro-particules sont déplacées en considérant uniquement les interactions gravitationnelles. Le champ de matière initial évolue ainsi jusqu'à $z = 0$. Ces simulations sont très utiles pour étudier les effets de la gravité à grande échelle. Cependant elles ne sont pas adaptées à notre utilisation : afin d'avoir la résolution et le volume requis, la simulation nécessiterait beaucoup trop de macro-particules pour être réalisable dans un temps raisonnable.

Les simulations hydrodynamiques fonctionnent de la même manière que les simulations N-corps. Elles incluent, en plus des macro-particules de matière noire, la physique baryonique présente dans le milieu galactique. Les baryons sont aussi représentés par des macro-particules. Afin de résoudre l'intérieur des galaxies, les macro-particules utilisées possèdent une masse plus faible que dans le cas

1. https ://www.python.org/
2. https ://github.com/igmhub/SaclayMocks

des simulations N-corps. En contrepartie, le volume simulé est plus petit. Dans le cas des simulations hydrodynamiques, la densité, la pression et la tempéature sont tracées dans chaque cellule. Certains effets astrophysiques, comme les supernovae ou les AGN peuvent aussi être présents. Ces simulations nécessitent encore plus de temps de calcul que les simulations à N-corps. Elles ne sont donc pas non plus adaptées à notre utilisation.

Ainsi, seuls les *champs aléatoires gaussiens* (GRF pour Gaussian Random Field) permettent de générer un grand volume avec une bonne résolution. Ce sont des champs qui en chaque point prennent une valeur aléatoire selon une statistique gaussienne. Une fois générés, il est possible de donner à ces champs n'importe quelle fonction de corrélation, en utilisant la transformation de Fourier. Ces champs sont utilisés notament dans les simulations à N-corps, afin de fournir les conditions Cependant, l'utilisation des GRF ne donne pas accès aux non linéaritées qui peuvent émerger dans l'évolution des simulations N-corps et hydrodynamiques. La seule information provient de la fonction de corrélation que l'on applique au GRF. Mais cela est entièrement suffisant pour l'utilisation que nous en avons dans ce manuscrit : nous générons un champ gaussien destiné à simuler le champ d'absorption Lyα et le champ de quasars, dont les fonctions d'auto corrélation et de corrélation croisée sont choisies afin de correspondre à ce qui est observé dans les données.

2 Construction des mocks

Dans cette section, nous détaillons comment les mocks sont générés. Nous présentons d'abord la génération des champs de densité, puis comment les quasars sont tirés à partir de ces champs de densité. Nous expliquons ensuite comment nous calculons la densité le long de la ligne de visée de chaque quasar. Enfin, nous présentons comment cette densité est transformée en fraction de flux transmis, et comment nous tirons les HCD.

2.1 Les champs de densité

La première étape dans la création des mocks est de générer les boîtes qui contiennent le champ de densité δ. D'abord, un GRF est généré dans une boite de $2560 \times 2560 \times 1536$ voxels, chaque voxel faisant $d_{cell}^3 = (2{,}19\, h^{-1}\,\mathrm{Mpc})^3$. Afin que le champ δ possède la bonne fonction de corrélation, une transformation de Fourier 3D [1] est appliquée sur la boîte, puis la boîte δ_k ainsi obtenue est multipliée par

$$\sqrt{\frac{P(k)}{d_{cell}^3}}\,, \tag{1.1}$$

où $P(k)$ est le spectre de puissance désiré. Il est ensuite possible d'obtenir la boîte δ dans l'espace réel grâce à une transformation de Fourier inverse de la boîte δ_k. Ce procédé garanti que le champ δ suive le spectre de puissance $P(k)$. Le GRF pourrait être tiré directement dans l'espace k. Dans ce cas, il nous faut tirer deux champs gaussiens : un pour la partie réelle, et un autre pour la partie imaginaire. La transformation de Fourier prenant moins de temps que la génération du champ aléatoire, nous préférons générer le champ dans l'espace réel plutôt que dans l'espace k. Dans la suite nous décrivons les différentes boîtes nécessaires à la construction des mocks : les boîtes champs utilsées pour tirer les quasars, la boîte utilisée pour créer l'absorption Lyα, ainsi que les boîtes de vitesse et de gradient de vitesse. Afin de garantir leur corrélation, toutes ces boîtes sont construites à partir de la même boîte initial δ_k.

1. Nous utilisons la librairie pyFFTW (https ://github.com/pyFFTW/pyFFTW), une adaptation python de la librairie FFTW (http ://www.fftw.org/).

Les quasars

Afin de construire un relevé de quasars corrélés, nous tirons les quasars selon le champ dans l'espace réel $\delta_{\rm QSO}$, construit à partir de δ_k. Une première solution serait de tirer les quasars dans les voxels dont le champ $\delta_{\rm QSO}$ est supérieur à un certain seuil. Cette solution produit une fonction de corrélation correcte aux grandes échelles, mais pas aux petites. En effet, comme montré dans **prov** les objets tirés aux endroits où la densité est supérieure à un certain seuil suivent la fonction de corrélation de la densité sous-jacente, avec un biais qui dépend du seuil choisi, si et seulement si la fonction de corrélation est petite devant 1. La fonction de corrélation ainsi obtenue est correcte, sauf pour les petites échelles pour lesquelles la fonction de corrélation est importante. Une solution alternative consiste à considérer que les quasars suivent une distribution log-normale. Ceci permet d'obtenir une meilleure corrélation aux petites échelles. Ce choix est souvent fait pour simuler des relevés de galaxies (Agrawal et al. 2017), et est en accord avec ce qui est observé dans les données (Clerkin et al. 2016). Nous optons donc pour cette seconde solution. Ainsi, les quasars sont tirés dans chaque voxel avec une probabilité

$$P \propto {\rm e}^{\delta_q} \,, \tag{1.2}$$

où δ_q est le champ de densité dans le voxel considéré. Comme montré par Coles et Jones (1991), afin que les quasars suivent la fonction de corrélation $\xi(r)$, le champ δ_q doit suivre la fonction de corrélation

$$\xi_q(r) = \ln(1+\xi(r)) \,. \tag{1.3}$$

Nous verrons section 2.2 que, de manière à obtenir un relevé synthétique de quasars dont le biais dépend de z, nous utilisons trois boîtes qui suivent des distributions log-normales, à des redshifts différents. La probabilité pour tirer les quasars dépend de l'interpolation entre ces 3 boîtes. Pour chacune des boîtes, nous partons du spectre de puissance de la matière $P_{matière}(k)$ à $z=0$, fourni par Camb (Lewis, Challinor et Lasenby 1999). Nous multiplions ensuite ce spectre de puissance par $(b_{\rm QSO}(z_i)G(z_i))^2$, où $i \in [1\,;2\,;3]$. A l'aide de la transformation de Fourier, nous calculons la fonction de corrélation $\xi_i(r)$. Puis, nous déterminons le spectre de puissance $P_{{\rm QSO},i}(k)$, à appliquer à la boîte δ_k, comme la transformée de Fourier de $\xi_{{\rm QSO},i}(r) = \ln(1+\xi_i(r))$ (équation **??**). Une fois les trois spectres de puissances $P_{{\rm QSO},i}(k)$ obtenus, nous construisons 3 boîtes

$$\delta_{k,i}(k) = \delta_k(k)\sqrt{\frac{P_{{\rm QSO},i}(k)}{V_{cell}}} \,, \tag{1.4}$$

où δ_k est le GRF dans l'espace de Fourier. Une fois ces 3 boîtes construites, nous appliquons à chacune d'entre elle une transformation de Fourier inverse afin d'obtenir les boîtes $\delta_{{\rm QSO},i}$. Ces boîtes seront interpolées en z, puis les quasars seront ensuite tirés avec une probabilité $\propto \exp(\delta_{\rm QSO}(z))$, où $\delta_{\rm QSO}$ est la boîte interpolée. Nous expliquons cette étape dans la section 2.2.

Le champ Lyα

Afin de construire le champ d'absorption Lyα, nous avons besoin du champ de densité de l'hydrogène neutre. Comme expliqué dans la section **??**, la fraction de flux transmis F est reliée à la profondeur obtique τ par

$$F = \exp(-\tau) \,. \tag{1.5}$$

De plus, la formule FGPA (Fluctuating Gunn Peterson Approximation) permet de relier la profondeur optique τ au contraste de densité δ à $z=0$:

$$\tau(z) = a(z)\exp(b(z)G(z)\delta) \tag{1.6}$$

Les paramètres a et b sont des paramètres à ajuster afin d'obtenir le bon biais du Lyα et la bonne transmission moyenne $\overline{F}$. Leur détermination est décrite dans la section 4. Le facteur de croissance G

prend en compte l'évolution avec le redshift du champ de densité δ. Ainsi il nous suffit de construire un GRF qui suit la fonction de corrélation à $z = 0$ pour simuler le champ d'absorption du Lyα. Pour ce faire, nous partons de la même boîte δ_k utilisée pour construire les 3 boîtes log-normales des quasars. Ceci garanti la corrélation croisée entre les quasars et le champ d'absorption Lyα. Le spectre de puissance de la matière $P_{matière}(k)$ à $z = 0$ est ensuite appliqué à la boîte δ_k. Enfin, nous obtenons la boîte de densité $\delta_{matière}$ à $z = 0$ qui servira au calcul du champ d'absorption du Lyα en effectuant la transformée de Fourier de la boîte

$$\delta_{k,matière}(\vec{k}) = \delta_k(\vec{k})\sqrt{\frac{P_{matière}(\vec{k})}{V_{cell}}} \,. \tag{1.7}$$

Les champs des vitesses

Afin d'inclure les RSD dans nos mocks, nous simulons aussi le champ des vitesses. A l'ordre linéaire, le champ des vitesses $v_{k,n}$ dans l'espace k selon la direction $\vec{u}_n$, avec $n \in [\mathrm{X};\mathrm{Y};\mathrm{Z}]$, est relié au champ de densité δ_k par la relation (voir équation 9.18 et 9.20 de Dodelson (2003))

$$v_{k,n}(\vec{k}) = \frac{ik_n}{k^2} aHf\delta_k(\vec{k}) \,. \tag{1.8}$$

En théorie linéaire, les traceurs sont considérés avoir la même vitesse que le champ de matière sous-jacent. Autrement dit, le champ de vitesse des traceurs est non biaisé. Nous adoptons ici cette hypothèse. **En ce qui concerne les quasars, nous simulons les RSD en déplaçant chaque quasar proportionnellement à sa vitesse le long de la ligne de visée (équation ??). Dans ce but, nous calculons les trois boîtes de vitesses $v_{k,\mathbf{x}}$, $v_{k,\mathbf{y}}$ et $v_{k,\mathbf{z}}$, comme**

$$v_{k,n}(\vec{k},z) = \frac{-ik_n}{k^2} H(z)\frac{dG}{dz}(z)\delta_{k,matière}(\vec{k}) \,; \quad n \in [\mathrm{X};\mathrm{Y};\mathrm{Z}] \,. \tag{1.9}$$

Ces boîtes sont construites pour $z = 0$. Cette équation est équivalente à l'équation 1.8, car

$$f = \frac{d\ln G}{d\ln a} = \frac{a}{G}\frac{dG}{da} = -\frac{1}{aG}\frac{dG}{dz} \tag{1.10}$$

Comme précédemment, la boîte $\delta_{k,matière}$ est la même que celle utilisée pour construire les boîtes $\delta_{\mathrm{QSO},i}$ des quasars et la boîte $\delta_{matière}$ utilisée pour le Lyα, ceci afin de garantir la correspondance entre la densité des traceurs et leurs vitesses particulières. A l'aide d'une transformation de Fourier inverse, nous obtenons les trois boîtes de vitesse à $z = 0$ dans l'espace réel v_X, v_Y et v_Z. Le calcul pour obtenir la vitesse parallèle $v_\parallel$ à un redshift z est décrit dans la section 2.2.

Concernant le champ d'absorption Lyα, les RSD sont prises en compte par une modification de la formule FGPA. Pour ce faire, nous avons besoin du gradient de vitesse $\eta_\parallel$ à $z = 0$. Nous utilisons la définition donnée dans l'équation 2.2 de Arinyo-i-Prats et al. (2015) :

$$\eta = -\frac{1}{aH}\frac{\partial v_p}{\partial x_p} \,, \tag{1.11}$$

où η est le gradient sans dimension de la vitesse v_p le long de la ligne de visée, et x_p est la coordonnées comobile le long de cette ligne de visée. En utilisant la définition précédente, et l'équation 1.8, nous obtenons le gradient η_{nm} de la vitesse v_n selon la direction $\vec{u}_m$ en fonction du champ δ_k :

$$\eta_{nm}(\vec{k}) = \frac{k_n k_m}{k^2} f\delta_k(\vec{k}) \,. \tag{1.12}$$

Cette équation permet de retrouver la formule de kaiser :

$$\begin{aligned} \delta_k^s(\vec{k}) &= \delta_k(\vec{k}) + \eta_\parallel(\vec{k}) \,, \\ &= (1 + f\mu_k^2)\delta_k(\vec{k}) \,. \end{aligned} \tag{1.13}$$

La boîte δ_k utilisée est le GRF initial, afin de garantir les corrélations entre les différents champs. A l'aide d'une transformation de Fourier inverse, nous obtenons [1] les 6 boîtes de gradients de vitesses à $z = 0$ dans l'espace réel η_{XX}, η_{YY}, η_{ZZ}, η_{XY}, η_{YZ} et η_{XZ}.

2.2 Le relevé de quasars

Une fois toutes ces boîtes construites, nous définissons la géométrie du relevé. Les boîtes, d'une taille $2560 \times 2560 \times 1536$ selon les axes X, Y et Z respectivement, sont placées à un redshift central $z_0 = 1{,}71$, et à une ascension droite α_0 et une déclinaison δ_0 (voir équation 1.20). Leurs dimensions permettent de couvrir les redshifts $1{,}3 < z < 3{,}6$. Cette limite basse est choisie de manière à pouvoir inclure les absorptions du CIV. Comme indiqué dans la section **??**, le pixel d'absorption de plus bas redshift dans la région Lyα se trouve à $z = 1{,}96$. Si cette absorption est causée par du CIV, l'absorbeur se situe à un redshift $z = (1+1{,}96)\lambda_{\mathrm{Ly}\alpha}/\lambda_{CIV} - 1 \sim 1{,}32$. Enfin, l'observateur est considéré être à $z = 0$, au centre du plan (X, Y).

Afin d'obtenir un biais des quasars qui dépend du redshift, nous pouvons modifier l'équation 1.2 et inclure une facteur $\alpha(z)$ qui prend en compte l'évolution avec le redshift du champ δ_q associé aux quasars :

$$P \propto \exp(\alpha(z)\delta_q)\,, \tag{1.14}$$

où $\alpha(z)$ est donné par

$$\alpha(z) = \frac{b_{\mathrm{QSO}}(z)(1+z_0)}{b_{\mathrm{QSO}}(z_0)(1+z)}\,, \tag{1.15}$$

et z_0 est le redshift auquel est construit le champ log-normal δ_q, avec b_{QSO} son biais. **La paramétrisation que nous utilisons pour le biais des quasars est donnée dans l'équation ??. La fonction de corrélation ξ_i à un redshift z_i est alors obtenue à partir de l'équation 1.3, et s'exprime comme**

$$\xi_i = \exp\Big(\alpha^2(z_i)\ln(1+\xi_0)\Big) - 1\,, \tag{1.16}$$

où ξ_0 est la fonction de corrélation à $z_0 = 2{,}33$, obtenue comme la transformation de Fourier du spectre de puissance $P_{matière}(k)$ à $z = 0$ fourni par Camb, et multipliée par le facteur de croissance G et le biais des quasars b_{QSO} au redshit z_0. Dans la limite où $\xi_0 \ll 1$ ou $|1-\alpha| \ll 1$, nous obtenons comme souhaité $\xi_i = \alpha^2(z_i)\xi_0$. La figure 1.1 compare les fonctions de corrélation $\xi_i/\alpha^2(z_i)$ avec la fonction de corrélation de référence ξ_0. Nous pouvons remarquer que plus le redshift z_i est éloigné de z_0, et plus l'écart entre $\xi_i/\alpha^2(z_i)$ et ξ_0 est grand. Ceci est d'autant plus vrai pour les petites séparations r, pour lesquelles ξ_0 n'est pas négligeable devant 1. Nous pouvons aussi remarquer sur le graphique de droite de la figure 1.1 que les formes des rapports $\xi_i/\alpha^2(z_i)\xi_0$ pour $z_i > z_0$ et $z_i < z_0$ sont semblables et de signe opposé. Nous pouvons donc combiner ces fonction de corrélation afin d'avoir un résultat plus satisfaisant.

Nous considérons donc maintenant deux champs log-normaux $\delta_{\mathrm{QSO},1}$ et $\delta_{\mathrm{QSO},2}$ à $z_1 = 2{,}1$ et $z_2 = 3{,}5$. Pour chaque champ, comme précédemment, nous construisons le champ

$$\hat{\delta}_{\mathrm{QSO},i}(z) = \exp(\alpha(z)\delta_{\mathrm{QSO},i})\,, \qquad i \in [1\,;2]\,, \tag{1.17}$$

puis, nous définissons le champ $\hat{\delta}_{\mathrm{QSO},12}$ comme une interpolation linéaire des deux champs définis dans l'équation précédente :

$$\hat{\delta}_{\mathrm{QSO},12}(z) = \hat{\delta}_{\mathrm{QSO},1}(z)\frac{z_2 - z}{z_2 - z_1} + \hat{\delta}_{\mathrm{QSO},2}(z)\frac{z - z_1}{z_2 - z_1}\,. \tag{1.18}$$

La figure 1.2 montre les fonctions de corrélation normalisées ξ_i qui correspondent au champ $\hat{\delta}_{\mathrm{QSO},12}$ à

1. Lors de la construction des 6 boîtes de gradients de vitesses, nous omettons volontairement le facteur $f(z = 0)$ de l'équation 1.12. Ce facteur f manquant sera pris en compte lors de l'ajout de la dépendance en redshift (voir section 2.4)

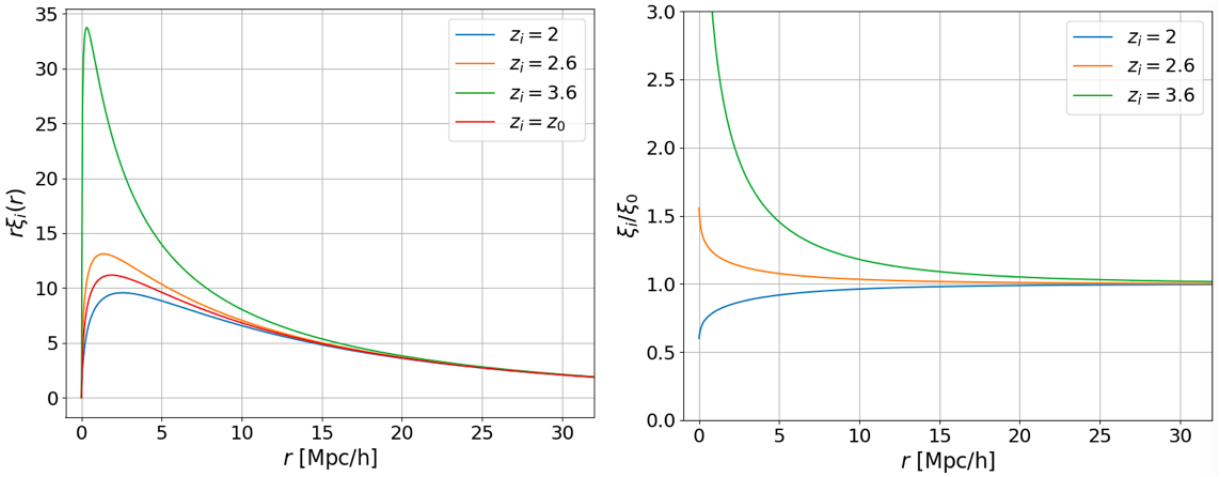

FIGURE 1.1 – Gauche : fonctions de corrélation ξ_i obtenues à partir de l'équation 1.16 et normalisées par un facteur $1/\alpha^2(z_i)$. La fonction de référence ξ_0 est calculée à un redshift $z_0 = 2{,}33$. Droite : rapports $\xi_i/\alpha^2(z_i)\xi_0$.

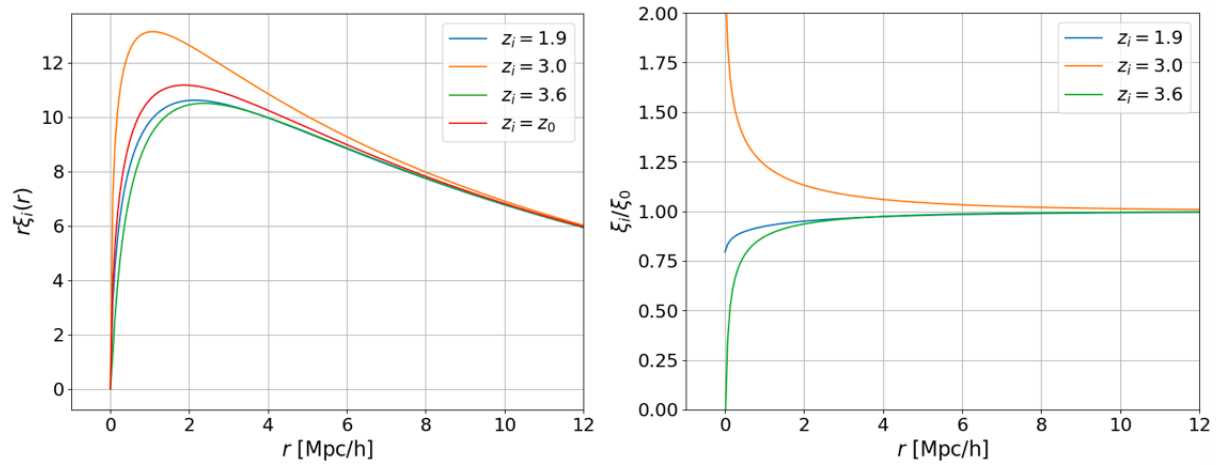

FIGURE 1.2 – Gauche : fonctions de corrélations $\xi_i(z_i)$, normalisées par un facteur $1/\alpha^2(z_i)$ et obtenues par interpolation (orange) et extrapolation (bleu et vert) des fonctions de corrélations ξ_1 et ξ_2 à $z_1 = 2{,}1$ et $z_2 = 3{,}5$. Leur obtention est détaillée dans le texte. Droite : rapports $\xi_i/\alpha^2(z_i)\xi_0$. Le redshift $z = 3{,}0$ est le redshift pour lequel la déviation par rapport à ξ_0 est la plus importante.

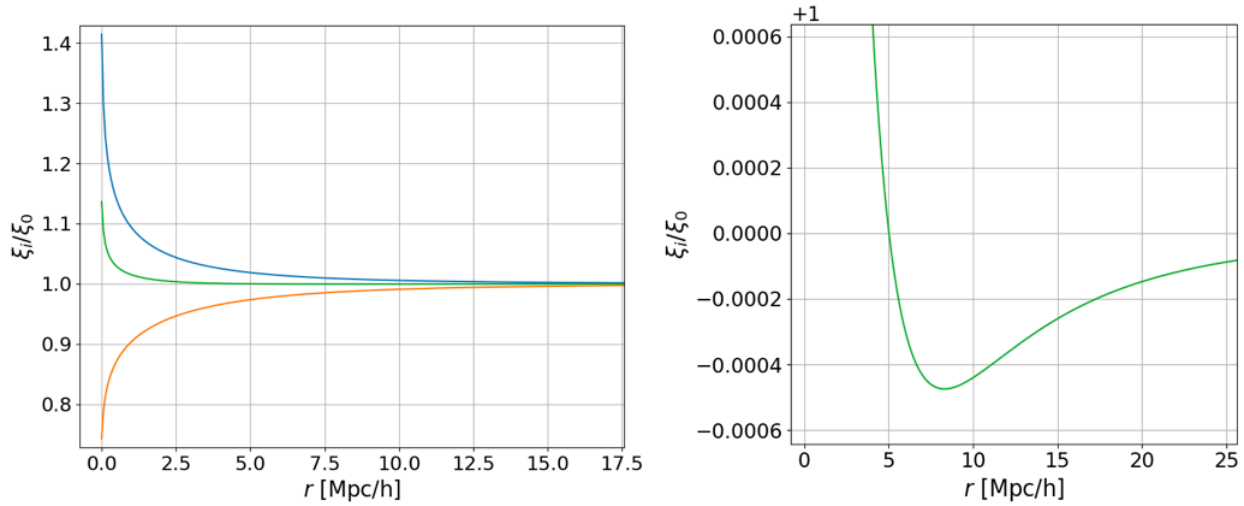

FIGURE 1.3 – Rapports normalisés $\xi_{12}(z)/\xi_0$ (bleu) et $\xi_{23}(z)/\xi_0$ (orange) à $z = 2{,}38$. Pour ce redshift, ξ_{12} est obtenue comme une interpolation de ξ_1 et ξ_2, et ξ_{23} comme une extrapolation de ξ_2 et ξ_3. La courbe verte est obtenue comme une combinaison linéaire de ξ_{12} et ξ_{23}. Le redshift $z = 2{,}38$ est le redshift pour lequel la courbe verte dévie le plus de ξ_0. Le détail est donné dans le texte. Le graphique de droite présente un zoom de celui de gauche.

différents redshifts z_i. Elles sont obtenues en interpolant linéairement (équation 1.18) les fonctions de corrélation ξ_1 et ξ_2, elles-mêmes obtenues aux redshifts z_1 et z_2 grâce aux transformations log-normales (équation 1.16) de $\alpha^2(z_1)\xi_0$ et $\alpha^2(z_2)\xi_0$. Nous pouvons remarquer sur cette figure 1.2 que l'écart entre les fonctions de corrélation ξ_1 et ξ_2 (en rouge) et les fonctions de corrélation interpolées est moins important que dans le cas avec un seul champ lognormal. La courbe orange montre la fonction de corrélation ξ_i au redshift $z_i = 3{,}0$ pour lequel l'écart avec ξ_1 et ξ_2 est maximal. Comme précédemment, nous pouvons constater que l'effet entre une interpolation (orange) et une extrapolation (vert, par exemple) et similaire et de signe opposé. Afin d'affiner encore l'évolution avec le redshift du champ utilisé pour tirer les quasars, nous pouvons considérer trois champs log-normaux, et combiner une interpolation et une extrapolation obtenues à partir de ces trois champs.

Nous considérons donc finalement trois champs log-normaux $\delta_{\mathrm{QSO},1}$, $\delta_{\mathrm{QSO},2}$ et $\delta_{\mathrm{QSO},3}$ à $z_1 = 1{,}9$, $z_2 = 2{,}75$ et $z_3 = 3{,}6$. C'est la solution qui a été choisie et implémentée dans les mocks. Ces trois champs sont stockés dans les boîtes construites dans la section 2.1. A partir de ces trois boîtes, nous construisons les 3 boîtes $\hat{\delta}_{\mathrm{QSO},1}$, $\hat{\delta}_{\mathrm{QSO},2}$ et $\hat{\delta}_{\mathrm{QSO},3}$ (équation 1.17), puis les deux boîtes interpolées $\hat{\delta}_{\mathrm{QSO},12}$ et $\hat{\delta}_{\mathrm{QSO},23}$ (équation 1.18). Enfin, nous construisons la boîte

$$\hat{\delta}_{\mathrm{QSO}}(z) = K(z)\hat{\delta}_{\mathrm{QSO},12}(z) + (1 - K(z))\hat{\delta}_{\mathrm{QSO},23}(z)\,, \tag{1.19}$$

comme la combinaison linéaire de deux boîtes interpolées $\hat{\delta}_{\mathrm{QSO},12}$ et $\hat{\delta}_{\mathrm{QSO},23}$. Le paramètre $K(z)$ est déterminé tel que, pour chaque z, le rapport de la fonction de corrélation ξ_{QSO}, normalisée par $1/\alpha^2(z)$ et obtenue avec l'équation 1.19, par la fonction de corrélation ξ_0, vaille 1 à $r = 5\,h^{-1}$ Mpc. La figure 1.3 présente les rapports normalisés $\xi_{12}(z)/\xi_0$ (correspondant à l'interpolation, en bleu) et $\xi_{23}(z)/\xi_0$ (correspondant à l'extrapolation, en orange) au redshift $z = 2{,}38$. La combinaison linéaire des deux (équation 1.19) est montrée en vert. Le redshift $z = 2{,}38$ correspond au redshift pour lequel la différence entre ξ_{QSO} et ξ_0 est la plus grande. Le graphique de droite de cette figure 1.3 montre un zoom de la courbe verte. Sur ce zoom, nous pouvons voir que ξ_{QSO} ne dévie pas de ξ_0 de plus de 5×10^{-4} pour $r > 5\,h^{-1}$ Mpc.

Les quasars sont ensuite tirés dans chaque voxel, avec une probabilité $P \propto \hat{\delta}_{\mathrm{QSO}}$. Pour ce faire, nous générons une variable ϕ aléatoire uniforme entre 0 et 1 dans chaque voxel. Les voxels pour lesquelles

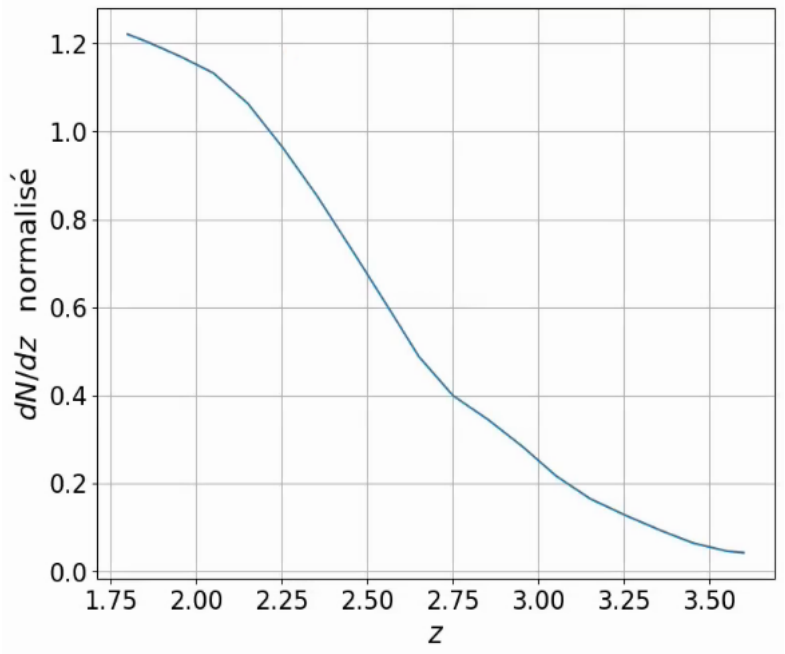

FIGURE 1.4 – Distribution normalisée en redshift des quasars tirés dans les mocks.

$\phi < N(z)\hat{\delta}_{\rm QSO}$ hébergent un quasar. Le facteur $N(z)$ contient l'évolution avec le redshift du nombre de quasars par degré carré. Il contient aussi un facteur de normalisation. Les quasars dont le redshift est en dehors de l'intervalle [1,8 ; 3,6] sont écartés. Les quasars dont l'ascension droite et la déclinaison sont en dehors des intervalles $[-\Delta\alpha\,;\Delta\alpha]$ et $[-\Delta\delta\,;\Delta\delta]$ sont aussi écartés. L'ascension droite α et la déclinaison δ du point $(\mathrm{X},\mathrm{Y},\mathrm{Z})$ sont données par

$$\alpha = \arctan\left(\frac{\cos(\alpha_0)\mathrm{X} - \sin(\delta_0)\sin(\alpha_0)\mathrm{Y} + \cos(\delta_0)\sin(\alpha_0)\mathrm{Z}}{-\sin(\alpha_0)\mathrm{X} - \sin(\delta_0)\cos(\alpha_0)\mathrm{Y} + \cos(\delta_0)\cos(\alpha_0)\mathrm{Z}}\right), \tag{1.20}$$

$$\delta = \arcsin\left(\frac{\cos(\delta_0)\mathrm{Y} + \sin(\delta_0)\mathrm{Z}}{\sqrt{\mathrm{X}^2+\mathrm{Y}^2+\mathrm{Z}^2}}\right). \tag{1.21}$$

Enfin, grâce au facteur $N(z)$, les quasars sont tirés selon la distribution en z prédite pour DESI. Cette distribution est présentée sur la figure 1.4. Cependant, nous tirons environ deux fois plus de quasars, afin de pouvoir simuler, entre autre, la sélection des cibles à l'aide du code `quickquasars` (présenté dans la section **??**). De plus, cela permet d'utiliser les mocks pour d'autres projets, comme WEAVE [1], qui possèdent une densité de cible plus élevée que celle de DESI. A la fin, nous obtenons environ 100 quasars à $z > 2{,}1$ par degré carré.

Une fois les quasars tirés, nous les déplaçons proportionnellement à leur vitesse $v_\parallel(z)$ le long de la ligne de visée. La vitesse, à $z = 0$, le long de la ligne de vitesse est obtenue comme

$$v_\parallel = \frac{v_{\rm X}\mathrm{X} + v_{\rm Y}\mathrm{Y} + v_{\rm Z}\mathrm{Z}}{\sqrt{\mathrm{X}^2+\mathrm{Y}^2+\mathrm{Z}^2}}, \tag{1.22}$$

où les vitesses $v_{\rm X}$, $v_{\rm Y}$ et $v_{\rm Z}$ sont les vitesses obtenus par transformation de Fourier inverse des vitesses $v_{k,{\rm X}}$, $v_{k,{\rm Y}}$ et $v_{k,{\rm Z}}$, définies dans l'équation 1.9. Ainsi, un quasar situé à une distance R sera replacé le

1. WEAVE est un spectrographe à fibre multi objet. Il possède 1000 fibres avec un champ de vue de 3,1 deg^2. Le relevé WEAVE-QSO prévoit d'observer 350 000 quasars à grand redshift, sur un relevé de 6000 deg^2 (PIERI et al. 2016).

long de la ligne de visée à une distance

$$R \to R + \frac{(1+z)}{H(z)} v_{\parallel}(z) \,, \tag{1.23}$$

où $v_{\parallel}(z)$ est relié à $v_{\parallel}$ à $z = 0$ par (voir équation 1.9)

$$v_{\parallel}(z) = \frac{H(z) dG/dz}{[H(z) dG/dz]_{z=0}} v_{\parallel} \,. \tag{1.24}$$

Le facteur $(1+z)$ vient de la conversion des distances en distances comobiles. Une fois tous les quasars déplacés, leur redshift est recalculé, puis ils sont stockés dans un catalogue. Pour chaque quasar, le catalogue contient leur position dans le ciel (α, δ), leur redshift avec et sans RSD, ainsi qu'un identifiant unique.

2.3 Création des lignes de visée

A cette étape, nous disposons d'un catalogue de quasars corrélés avec le champ de densité $\delta_{matière}$ qui sera utilisé pour construire l'absorption Lyα. Nous pouvons donc créer les lignes de visées à partir de chaque quasar et jusqu'à l'observateur, puis interpoler la boîte contenant le champ de densité le long de ces lignes de visée. Dans un premier temps, nous commençons par définir le tableau en longueurs d'onde observées, sur lequel sera interpolé la boîte de densité. Nous choisissons une taille de pixel $d_{pix} = 0{,}2\,h^{-1}$ Mpc. Les limites $1{,}8 < z < 3{,}6$ en redshift se traduisent par des limites $3403{,}876 < \lambda < 5592{,}082\,$Å sur la longueur d'onde observée pour le Lyα. Nous ajoutons la limite basse des spectrographes de DESI : $\lambda_{min} = 3530\,$Å, que nous réduisons afin d'inclure certains métaux dans les forêts (cela sera expliqué dans la section **??**). Les longueurs d'onde observées couvrent donc $3476{,}1877 < \lambda < 5591{,}566\,$Å à l'aide de 6524 pixels [1].

Une fois ce tableau en longueur d'onde défini, nous faisons correspondre le pixel $\lambda_{\rm QSO} = (1 + z_{\rm QSO})\lambda_{{\rm Ly}\alpha}$ à la position $(\mathrm{X_{QSO}}, \mathrm{Y_{QSO}}, \mathrm{Z_{QSO}})$ du quasar. Puis, pour chaque pixel i entre $\lambda_{min} = 3476{,}1877$ et $\lambda_{\rm QSO}$, la position $(\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i)$ du pixel est déterminée. Pour chaque pixel, nous calculons une moyenne pondérée des voxels voisins avec un lissage gaussien. Ce lissage est nécessaire afin d'éviter le crénelage (*aliasing*) aux petites échelles. Sans lissage, les spectres interpolés possèdent des discontinuitées (voir figure 1.5). Ces discontinuités rajoutent de la puissance parasite aux petites échelles lors du calcul du spectre de puissance. Nous appliquons donc un lissage gaussien, de largeur $\sigma = d_{cell}$. La figure 1.5 présente le champ interpolé en considérant uniquement les voxels à $\pm 1\sigma$. Trop peu de voxels sont considérés pour le calcul de chaque pixel, et le champ obtenu contient des discontinuités. La figure 1.6 présente le champ interpolé en considérant les voxels à $\pm 2\sigma$ (orange) et $\pm 3\sigma$ (bleu). La différence entre les deux est faible, cependant certaines discontinuités subsistent pour le champ calculé avec 2σ, comme le montre le zoom sur le graphique de droite. **Enfin, nous avons vérifié que les différences entre les champs obtenus avec 3 et 4σ sont plus faibles que 10^{-2}.** Nous nous limitons donc aux voxels compris à $\pm 3\sigma$ du voxel central pour limiter le temps CPU. Pour chaque pixel, nous considérons les voxels appartenant au cube de 7 voxels de côté, centré sur le voxel dans lequel se trouve le pixel. Ceci représente donc, pour chaque pixel, une moyenne sur 343 voxels. Le champ dans le pixel i est alors donné par

$$\delta_i = \frac{\sum_{j=0}^{342} \delta_j w_{ij}}{\sum_{j=0}^{342} w_{ij}} \,, \tag{1.25}$$

1. Ces limites sont aussi choisies afin de garantir un nombre entier de pixels.

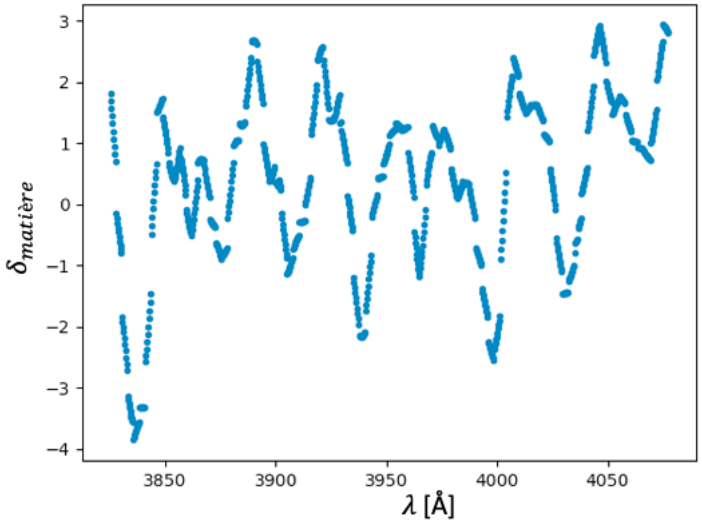

FIGURE 1.5 – Exemple de champ interpolé en utilisant les voxels à $\pm 1\sigma$ du voxel central. Le champ ainsi interpolé possède de nombreuses discontinuitées. Ces discontinuitées produisent de la puissance supplémentaire aux petites échelles (effet de crénelage ou d'*aliasing*).

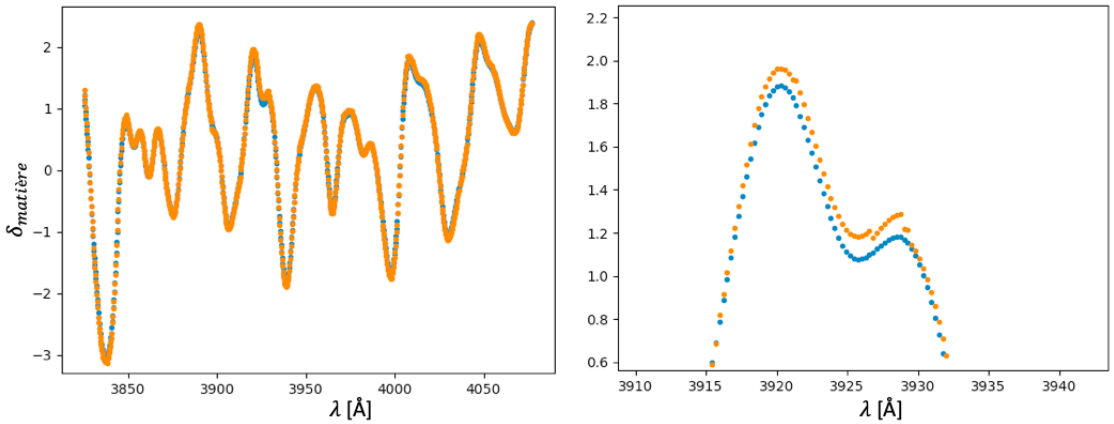

FIGURE 1.6 – Exemple de champ interpolé en utilisant les voxels à $\pm 2\sigma$ (orange) et à $\pm 3\sigma$ (bleu) du voxel central. Les discontinuitées sont beaucoup moins nombreuses que pour le champ interpolé avec $\pm 1\sigma$ (figure 1.5). Cependant, certaines sont encore visibles sur le zoom, présenté sur le graphique de droite.

avec

$$w_{ij} = \exp\left(\frac{-(\vec{r}_j - \vec{r}_i)^2}{2\sigma^2}\right), \tag{1.26}$$

où $\vec{r}_j$ est la position du centre du voxel j, $\vec{r}_i$ celle du pixel i, et δ_j est la valeur de la boîte au pixel j.

Ce calcul est effectué pour tous les pixels qui vérifient $\lambda_i < \lambda_{\rm QSO}$, pour chaque quasar. Les boîtes interpolées sont la boîte $\delta_{matière}$ utilisée pour construire l'absorption Lyα, les trois boîtes de vitesse utilisées pour ajouter les RSD aux HCD tirés dans chaque ligne de visée (voir section 2.5), et les six boîtes de gradient de vitesse afin d'ajouter les RSD au champ Lyα.

2.4 De la densité à l'absorption

Une fois les lignes de visés interpolées, nous pouvons transformer le champ de densité en absorption Lyα. Ceci est fait via la formule FGPA :

$$F = \exp\left[-a(z)\exp(b(z)G(z)\delta_{matière})\right] . \tag{1.27}$$

Les petites échelles

La boîte $\delta_{matière}$, utilisée dans l'équation 1.27, contient le champ de matière à grande échelle. Elle est construite grâce à la transformation de Fourier de la boîte δ_k. Ce champ est construit sur une grille d'intervalle $d_{cell} = 2{,}19\,h^{-1}$ Mpc. Par conséquent, la plus petite échelle accessible est

$$k_N = \frac{\pi}{d_{cell}} \sim 1{,}43\,h \text{ Mpc}^{-1} . \tag{1.28}$$

Nous manquons donc toutes les fluctuations pour lesquelles $k > k_N$. **Ces fluctuations ne sont pas importantes pour la corrélation à trois dimensions. La figure 1.7 montre le spectre de puissance de la matière $P_{matière}(k)$ à $z = 0$ (noir), obtenu avec Camb, ainsi que l'effet du lissage gaussien appliqué aux voxels et l'effet de la taille non nulle de ces voxels. La courbe bleu donne le spectre de puissance qui inclut l'effet du lissage gaussien. Il s'exprime comme**

$$P(k) = W^2(k)P_{matière}(k) , \tag{1.29}$$

où $W(k)$ est le terme représentant le lissage gaussien appliqué à l'interpolation de la boîte δ_l :

$$W(k) = \exp\left(-\frac{kd_{cell}}{2}\right) . \tag{1.30}$$

La courbe verte donne le spectre de puissance qui inclut l'effet de la taille des voxels. Celui-ci s'exprime comme

$$P(k) = G^2(k)P_{matière}(k) , \tag{1.31}$$

où $G(k)$ est le terme qui prend en compte la taille des voxcels :

$$G(k) = \text{sinc}(\frac{kd_{cell}}{2}) . \tag{1.32}$$

La courbe rouge donne le spectre de puissance qui inclut ces deux effets, c'est à dire multiplié par $W^2(k)G^2(k)$. La figure 1.8 montre la transformation de Fourier de ces spectres de puissance. Les effets liés au lissage gaussien et à la taille des voxels sont principalement localisés à petit r et autour du pic BAO. Ces effets sont faibles, en particulier celui produit par la taille non nulle des voxels. Ainsi, ces fluctuations à petite échelle ne sont pas importantes pour la fonction de corrélation à trois dimension. Cependant, ces fluctuations sont importantes pour le spectre de puissance à une dimension. Celui-ci donne

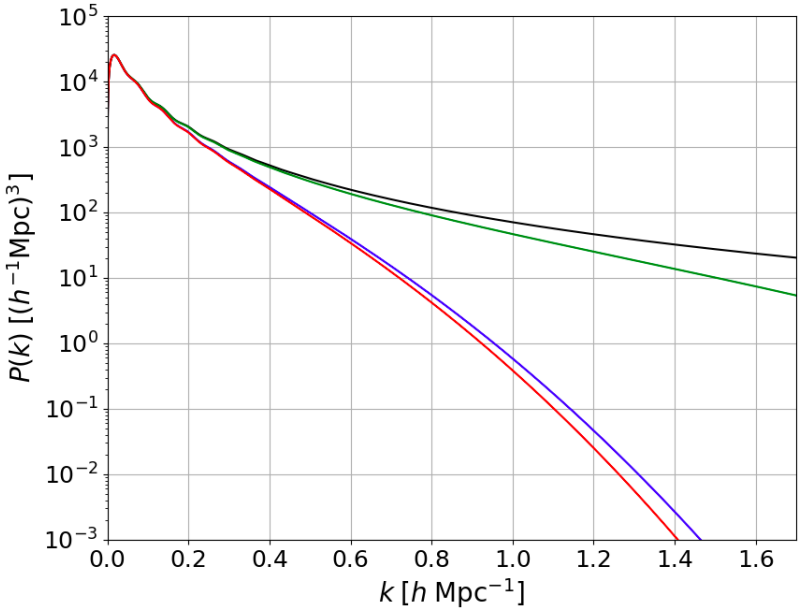

FIGURE 1.7 – Effet du lissage gaussien et de la taille non nulle des pixels sur le spectre de puissance. La courbe noire donne le spectre de puissance de la matière, obtenu avec Camb. La courbe bleu donne le spectre de la matière multiplié par $W^2(k)$ (équation 1.30). La courbe verte donne le spectre de puissance multiplié par $G^2(k)$ (équation 1.30). Enfin la courbe rouge donne le spectre de puissance multiplié par $W^2(k)G^2(k)$.

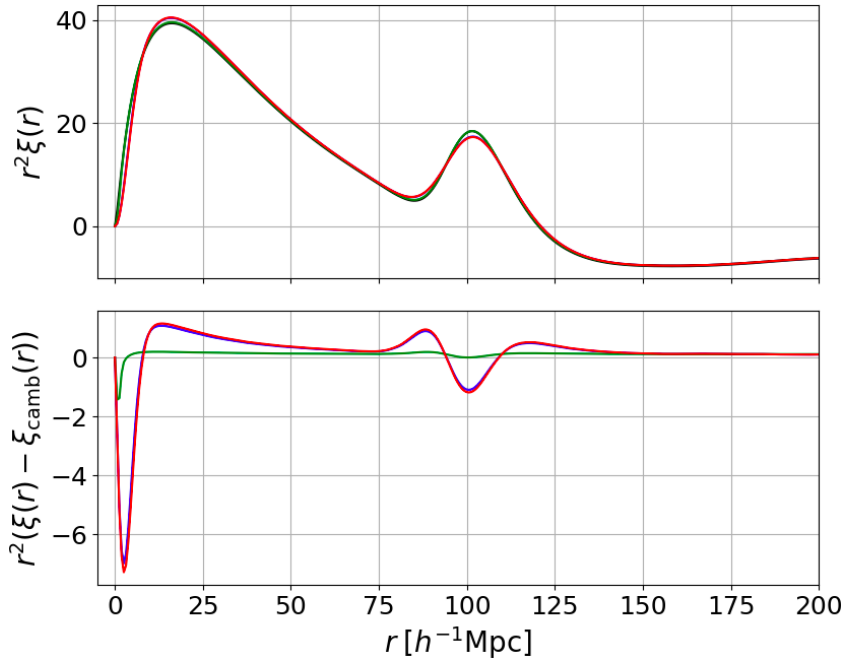

FIGURE 1.8 – Effet du lissage gaussien et de la taille non nulle des pixels sur la fonction de corrélation des mocks. Le graphique du haut montre les fonctions de corrélation obtenues à partir du spectre de puissance de Camb (noir), du spectre de puissance de Camb multiplié par $W^2(k)$ en bleu, du spectre de puissance de Camb multiplié par $G^2(k)$ en vert, et du spectre de puissance de Camb multiplié par $W^2(k)G^2(k)$ en rouge (voir équations 1.30 et 1.32). Le graphique du bas montre la différence entre les courbes bleue et noire, en bleu, entre les courbes vert et noire, en vert, ainsi qu'entre les courbes rouge et noire, en rouge.

la puissance mesurée le long de la ligne de visée. Pour un mode $k_\parallel$ donné, il correspond à l'intégrale du spectre de puissance à trois dimensions sur la direction transverse à la ligne de visée :

$$P^{\mathrm{1D}}(k_\parallel) = \frac{1}{2\pi}\int_0^\infty k_\perp P(k_\parallel, k_\perp) dk_\perp \,. \tag{1.33}$$

Avec la taille non nulle des voxels, l'intégrale s'effectue seulement jusqu'à k_N. De plus, le lissage gaussien appliqué à l'interpolation de la boîte δ_l réduit davantage l'amplitude de ce spectre de puissance. Ces deux effets sont représentés sur la figure ?? (#prov faire la figure). Contrairement au cas à trois dimensions, l'effet de la taille des voxels et du lissage gaussien est important pour le spectre de puissance à une dimension. En effet, ce spectre de puissance le long d'une ligne de visée est beaucoup plus sensible au bruit que le spectre de puissance calculé sur une colonne de section de $(2{,}19\,\boldsymbol{h}^{-1}\,\mathrm{Mpc})^2$. A cause de ces deux effets, l'amplitude du spectre de puissance à une dimension n'est pas correcte. Et ainsi, la variance du champ, donnée par l'intégrale du P^{1D}, n'est pas correcte non plus. Pour palier ce problème, nous rajoutons indépendamment sur chaque ligne de visée un champ δ_s (*small scales* : petites échelles) au champ $\delta_{matière}$ que nous appelons désormais δ_l (*large scales* : grandes échelles) :

$$F = \exp\left[-a(z)\exp(b(z)G(z)(\delta_l + \delta_s))\right];. \tag{1.34}$$

Comme ce champ n'est pas corrélé d'une ligne de visée à une autre, il ne participe pas à la fonction de corrélation à trois dimensions [1]. Afin d'ajouter la bonne quantité de fluctuations aux petites échelles, pour chaque ligne de visée nous générons un GRF à une dimension $\delta_{k,s}$. Puis, nous multiplions ce champ par

$$\sqrt{\frac{P_s(k, z_{\mathrm{eff}})}{d_{pix}}}\,, \tag{1.35}$$

où z_{eff} est le redshift moyen de la forêt, et P_s est le spectre de puissance qu'il faut appliquer à $\delta_{k,s}$ afin que δ_F possède le bon P^{1D}. $\delta_{k,s}$ est créé avec une taille supérieure à celle de la forêt, pour éviter les corrélations d'une extrémité à l'autre. La détermination de P_s est détaillée dans la section 4. Puis, nous obtenons δ_s à l'aide de la transformation de Fourier inverse de $\delta_{k,s}$. Avant d'ajouter δ_s à δ_l, nous corrigeons la dépendance en z de chacun des δ_s de la forêt. En effet, δ_s est construit de façon à obtenir le bon spectre de puissance à une dimension au redshift moyen de la forêt. Les pixels situés en bord de forêt n'auront donc pas le bon P^{1D}. Ainsi, nous corrigeons chaque δ_s comme :

$$\delta_s(z) \to \delta_s(z)\frac{\sigma_s(z)}{\sigma_s(z_{\mathrm{eff}})}\,, \tag{1.36}$$

où σ_s est l'écart type du champ δ_s. Celui-ci est relié au spectre de puissance P_s comme :

$$\sigma_s = \frac{1}{d_{pix}N}\left(P_s(0) + P_s(k_{Ny}) + 2\sum_{j=1}^{\frac{N}{2}-1} P_s\left(\frac{2\pi}{d_{pix}N}j\right)\right)\,, \tag{1.37}$$

où $k_{Ny} = \pi/d_{pix} \sim 15{,}7\,\boldsymbol{h}^{-1}\,\mathrm{Mpc}$ est le mode de Nyquist : c'est le mode maximal accessible pour une résolution donnée.

1. Lors du calcul de la fonction de corrélation à trois dimensions, nous ne considérons pas les paires de pixels issues de la même forêt (voir **??**)

Les RSD

Une fois les petites échelles ajoutées, nous obtenons un champ d'absorption F qui possède le bon spectre de puissance à trois dimensions, ainsi que le bon spectre de puissance à une dimension. Cependant, le champ d'absorption F ne possède pas de RSD pour l'instant, car il est construit à partir du spectre de puissance $P_{matière}(k)$ qui est isotrope. Initiallement, nous pensions ajouter les RSD au niveau du spectre de puissance : multiplier le GRF intial δ_k par $(1+\beta\mu^2)P_{matière}(k,\mu)$, avec $\mu = k_z/k$. Mais cette solution n'est pas envisageable car les lignes de visées ne sont pas parallèles (elles l'étaient pour les mocks précédents, développés pour BOSS), et nous ne pouvons donc pas confondre l'axe k_z avec l'axe de la ligne de visée $k_\parallel$. Une autre solution est d'utiliser le champ de vitesse défini dans l'équation 1.8. Chaque pixel d'absorption est alors déplacé proportionnellement à la vitesse particulière du gaz dans la cellule considérée. Puis l'absorption est modifiée en fonction de la différence des vitesses des pixels voisins. En effet, si cette différence est non nulle, le gaz se retrouve comprimé par endroit, et détendu dans d'autres, modifiant l'absorption dans chaque pixel. Cette méthode pour ajouter les RSD dans les mocks Lyα est la méthode choisie par Le Goff et al. (2011) et Farr et al. (2019). Ce n'est pas la solution que nous utilisons ici. Afin d'inclure les RSD dans le champ d'absorption F, nous utilisons le champ de gradient de vitesse $\eta_\parallel$, présenté dans la section 2.1. **Contrairement à la méthode qui utilise le champ de vitesse, la solution qui ajoute le champ $\eta_\parallel$ au champ δ_l permet de prédire la fonction de corrélation des mocks. Nous décrivons cette prédiction dans la section suivante.**

Plusieurs essais ont été menés afin de savoir comment inclure correctement le champ $\eta_\parallel$ dans FGPA. La première solution envisagée était de modifier la profondeur optique τ. Comme $\delta_\tau = \tau/\overline{\tau} - 1$ est conservée lors du passage dans l'espace des redshifts, nous modifiions τ (équation 1.6) comme

$$\tau \to \tau + \overline{\tau}\eta_\parallel \, . \tag{1.38}$$

Cette solution produisait une trop faible quantité de RSD. Nous avons alors décidé d'inclure les RSD causées par les petites échelles. Similairement à δ_l et δ_s, nous ajoutions à la profondeur optique la contribution aux grandes et petites échelles η_l et η_s. En faisant ceci, nous étions plus proche de ce que nous aurions obtenu avec une taille de voxel plus petite. La profondeur optique était alors donnée par

$$\tau = a\exp\left[bG(\delta_l + \delta_s)\right] + \overline{\tau}(\eta_l + \eta_s) \, . \tag{1.39}$$

Cette solution produisait la bonnne quantité de RSD. Cependant, la profondeur optique était négative pour un nombre non négligeable de pixels, résultant en une absorption $F > 1$ non physique pour ces pixels. Finalement, nous avons décidé d'inclure le gradient de vitesse directement au niveau du champ δ. Le champ $\eta_\parallel$ est donc ajouté, en plus du champ δ_s, au champ δ_l. Ceci nous permet de retrouver la formule de Kaiser (équation 1.13). De plus, cette solution ne produit pas de pixel avec $F > 1$. Afin de gérer la bonne quantité de RSD, nous ajoutons un coefficient c, qui dépend de z. L'ajustement de ce paramètre nous permet d'obtenir la bonne dépendance en z pour $\beta_{\mathrm{Ly}\alpha}$. La formule FGPA devient donc

$$F = \exp\Big[-a(z)\exp\Big(b(z)G(z)(\delta_l + \delta_s + c(z)\eta_\parallel)\Big)\Big] ;. \tag{1.40}$$

Les champs δ_l, δ_s et $\eta_\parallel$ sont calculés à $z = 0$. De plus, le facteur f que nous avons laissé de côté dans la section 2.1 n'est pas explicité ici : pour les redshifts $z > 2$, l'univers est dominé par la matière et donc, en bonne approximation, nous avons $f(z) \sim 1$. Les paramètres $a(z)$, $b(z)$, $c(z)$, ainsi que $P_s(k,z)$ sont ajustés afin d'obtenir les bons $b_{\mathrm{Ly}\alpha}(z)$, $\beta_{\mathrm{Ly}\alpha}(z)$, $\overline{F}(z)$ et $P^{\mathrm{1D}}_{\mathrm{Ly}\alpha}(k,z)$. L'ajustement est décrit dans la section 4.

La prédiction

La méthode que nous utilisons pour ajouter les RSD, décrite dans la section précédente, a l'avantage d'avoir une fonction de corrélation prédictible.Dans les lignes qui suivent, nous expliquons comment

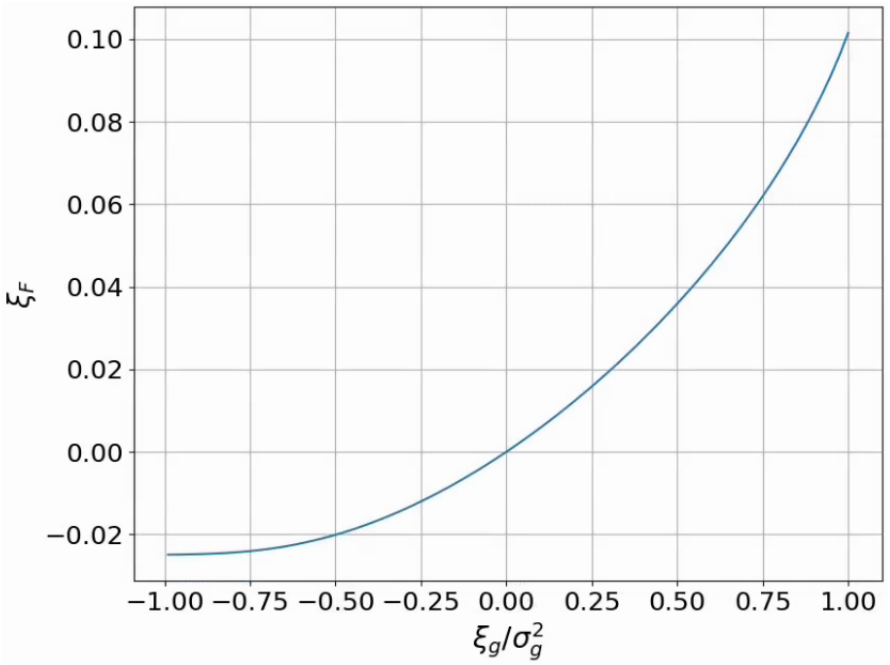

FIGURE 1.9 – La fonction de corrélation ξ_F obtenue à l'aide de l'équation 1.41 pour chaque valeur de $\xi_g/\sigma_g^2 \in [-1\,;1]$.

calculer la prédiction de la fonction de corrélation. Comme décrit par FONT-RIBERA, MCDONALD et MIRALDA-ESCUDÉ (2012), dans le cas où le champ δ_F est une fonction d'un champ gaussien δ_g, il est possible de relier la fonction de correlation ξ_F du champ δ_F à la fonction de corrélation ξ_g du champ δ_g. La fonction de corrélation $\xi_F(r_{12})$ pour la séparation r_{12} est reliée à $\xi_g(r_{12})$ par

$$\xi_F(r_{12}) = \int_{-\infty}^{\infty} d\delta_{g1} \int_{-\infty}^{\infty} d\delta_{g2} \frac{\exp\left[-\frac{\delta_{g1}^2+\delta_{g2}^2-2\delta_{g1}\delta_{g2}\xi_g(r_{12})}{2(1-\xi_g^2(r_{12}))}\right]}{2\pi\sqrt{1-\xi_g^2(r_{12})}} \delta_F(\delta_{g1})\delta_F(\delta_{g2})\,, \tag{1.41}$$

où δ_g est un GRF de variance 1 et δ_F est le champ d'absorption calculé à partir du champ gaussien. Dans notre cas, le champ δ_g représente le champ $\delta_l + \delta_s + c(z)\eta_\parallel$. Ce champ est un champ gaussien, de valeur moyenne nulle et de variance σ_g^2. Cette variance est dominé par la variance de δ_s. Nous compensons le fait que $\sigma_g^2 \neq 1$ en remplaçant ξ_g par ξ_g/σ_g^2 dans l'équation 1.41. Dans cette équation, $\xi_F(r_{12})$ ne dépend que de la valeur de $\xi_g(r_{12})/\sigma_g^2$. Nous construisons donc une table qui permet de relier chaque valeur de $\xi_g/\sigma_g^2 \in [-1\,;1]$ à la valeur ξ_F correspondante. **La figure 1.9 montre la fonction de corrélation ξ_F obtenue, en fonction de ξ_g/σ_g^2. Par ailleurs, puisque δ_s n'est pas corrélé d'une forêt à une autre, il ne participe pas au spectre de puissance à trois dimensions. Le champ δ_g suit alors le spectre de puissance**

$$P_g(k) = (1 + c\mu_k^2)^2 P_l(k)\,, \tag{1.42}$$

où P_l est le spectre de puissance que suit le champ δ_l. Afin de relier la fonction de corrélation dans l'espace des redshifts $\xi_g(r,\mu)$ que suit le champ δ_g à la fonction de corrélation $\xi(r)$ que suit le champ δ_l, nous utilisons les formules données dans Hamilton (1992) :

$$\xi_g(r,\mu) = \xi_0(r) + P_2(\mu)\xi_2(r) + P_4(\mu)\xi_4(r)\,, \tag{1.43}$$

avec

$$\xi_0(r) = \left(1 + \frac{2}{3}\beta + \frac{1}{5}\beta^2\right)\xi(r)\,, \tag{1.44}$$

$$\xi_2(r) = \left(\frac{4}{3}\beta + \frac{4}{7}\beta^2\right)\left[\xi(r) - \overline{\xi}(r)\right]\,, \tag{1.45}$$

$$\xi_4(r) = \frac{8}{35}\beta^2\left[\xi(r) + \frac{5}{2}\overline{\xi}(r) - \frac{7}{2}\overline{\overline{\xi}}(r)\right]\,, \tag{1.46}$$

et

$$\overline{\xi}(r) = 3r^{-3}\int_0^r \xi(s)s^2 ds\,, \tag{1.47}$$

$$\overline{\overline{\xi}}(r) = 5r^{-5}\int_0^r \xi(s)s^4 ds\,. \tag{1.48}$$

Les $P_l(\mu)$ sont les polynômes de Legendre :

$$P_0(\mu) = 1\ ; \qquad P_2(\mu) = \frac{1}{2}(3\mu^2 - 1)\ ; \qquad P_4(\mu) = \frac{1}{8}(35\mu^4 - 30\mu^2 + 3)\,, \tag{1.49}$$

et β est le paramètre RSD du Lyα. **Afin d'obtenir la prédiction $\xi_F^{pred}(r,\mu)$, nous commençons par calculer le spectre de puissance que suit la boîte δ_l :**

$$P(k) = W^2(k)P_{matière}(z=0)\,, \tag{1.50}$$

où $W(k)$ est le terme représentant le lissage gaussien appliqué à l'interpolation de la boîte δ_l (équation 1.30). Nous négligeons ici l'effet de la taille non nulle des voxels sur le spectre de puissance (équation 1.32). La figure 1.8 montre l'effet de $W(k)$ et $G(k)$ sur la fonction de corrélation. L'effet lié à $G(k)$ est nettement moins important que celui lié à $W(k)$. Puis, à l'aide d'une transformation de Fourier du spectre de puissance, nous obtenons la fonction de corrélation $\xi(r)$ que suit le champ δ_l obtenu par interpolation avec lissage des boîtes. Nous calculons ensuite la fonction de corrélation dans l'espace des redshifts $\xi_g(r,\mu)$ que suit le champ δ_g grâce à l'équation 1.43. La figure **??** (#prov faire la figure) présente cette fonction de corrélation mesurée sur une réalisation. Enfin, pour tous les couples (r,μ) nécessaires, nous obtenons la fonction de corrélation $\xi_F(r,\mu)$ du champ δ_F comme la valeur correspondante à la valeur tabulée $\xi_g(r,\mu)/\sigma_g^2$ pour ξ_g.

2.5 Ajout des HCD

Les HCD ont un effet important sur les fonctions de corrélation, nous simulons donc aussi leur présence. De manière à avoir une corrélation entre les HCD et les autres traceurs des mocks, nous utilisons la boîte de densité δ_l pour tirer les HCD. Nous ne considérons pas la somme $\delta_l + \delta_s$ car les HCD sont des surdensités qui se situent dans les structures à grandes échelles : une résolution de $2{,}19\,h^{-1}$ Mpc est suffisante. De plus, l'ajout de δ_s bruiterait la corrélation entre les HCD et les autres traceurs. En effet, l'écart type σ_s du champ δ_s est entre 4 et 6 fois plus important que celui du champ δ_l, et L'écart type σ_g est dominé à plus de 95 % par σ_s. Du fait que δ_s ne possède pas de corrélation à 3 dimensions, les HCD se situeraient principalement dans des pics de bruit non corrélés.

Contrairement aux quasars, les HCD sont tirés selon les pics du champ δ_l : nous identifions les pixels dans lesquelles δ_l est au dessus d'un certain seuil, puis les HCD sont tirés dans ces pixels selon une loi de Poisson. Le seuil ν est défini en fonction du biais souhaité pour les HCD. L'appendice A de Font-Ribera et Miralda-Escudé (2012) relie le rapport b_ν/b_g au seuil ν :

$$\left(\frac{b_\nu}{b_g}\right)^2 = \frac{p_g(\nu)}{\left(\int_\nu^\infty d\delta_g p_g(\delta_g)\right)^2}\int_\nu^\infty dy_1 p_g(y_1)y_1\,, \tag{1.51}$$

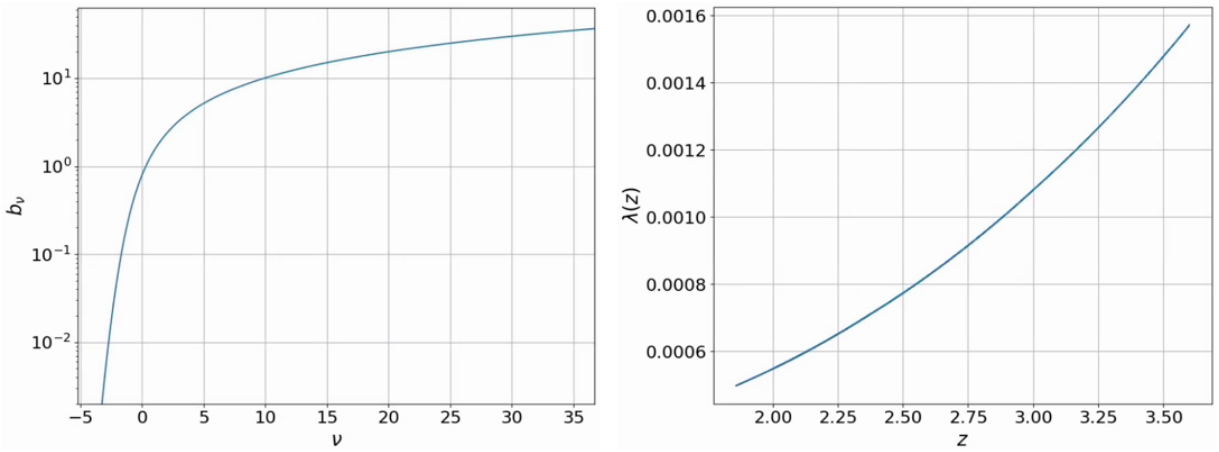

FIGURE 1.10 – Le graphique de gauche montre le biais b_ν obtenu pour un seuil ν. Le graphique de droite montre le paramètre $\lambda(z)$, utilisé pour tirer les HCD.

où b_ν est le biais obtenu avec le seuil ν, b_g est le biais du champ gaussien, et p_g donne la densité de probabilité du champ gaussien de variance 1. Dans notre cas, $b_g = 1$. **En utilisant le changement de variable $u = y_1^2/2$, l'intégrale au dénominateur de l'équation 1.51 vaut simplement $p_g(\nu)$:**

$$\int_\nu^\infty dy_1 p_g(y_1) y_1 = \int_\nu^\infty dy_1 \frac{\exp(-y_1^2/2)}{\sqrt{2\pi}} y_1 = \int_{\nu^2/2}^\infty du \frac{\exp(-u)}{\sqrt{2\pi}} = \frac{\exp(-\nu^2/2)}{\sqrt{2\pi}} = p_g(\nu)\,. \tag{1.52}$$

L'équation 1.51 est donc équivalente à

$$b_\nu = \frac{p_g(\nu)}{\int_\nu^\infty d\delta_g p_g(\delta_g)}\,. \tag{1.53}$$

Afin d'obtenir le seuil pour un biais donné, nous tabulons b_ν pour une large gamme de seuils ν, puis nous interpolons ν en fonction de b_ν. Le graphique de gauche de la figure 1.10 montre le biais b_ν en fonction du seuil ν. Dans notre cas, le champ δ_l suit une distribution de probabilité gaussienne. Cependant, sa variance n'est pas égale à 1. De plus, le champ δ_l interpolé le long des lignes de visée correspond au champ de matière à $z = 0$. Ainsi, pour obtenir un biais $b_{\rm HCD}$, pour chaque redshift nous calculons le seuil ν comme si nous visions un biais $b = b_{\rm HCD}\sigma_l G(z)$. Le terme σ_l prend en compte la variance du champ δ_l, et $G(z)$ le fait que δ_l soit construit à $z = 0$. Une fois les pixels pouvant héberger un HCD identifiées, nous tirons dans chacune d'entre elles les HCD avec une loi de poisson de paramètre

$$\lambda(z) = \frac{N(z)}{cdf(-\nu(z))}\,, \tag{1.54}$$

où $N(z)$ donne le nombre moyen de HCD attendu par pixel et $\nu(z)$ le seuil au redshift z. Le nombre de HCD attendu est donné par la librairie `pyigm` [1]. La distribution en redshift des HCD est présentée sur le graphique de gauche de la figure 1.11, et le paramètre $\lambda(z)$ sur le graphique de droite de la figure 1.10. En pratique, du fait que $\lambda(z) \ll 1$, il est très rare d'avoir plus d'un HCD par pixel. Une fois tous les HCD tirés, nous leur assignons une densité de colonne dans la gamme $17{,}2 < \log(n_{\rm HI}) < 22{,}5$, selon la distribution donnée par `pyigm`. Cette distribution est présentée sur le graphique de droite de la figure 1.11. Enfin, nous ajoutons les RSD aux HCD tirés. Chaque HCD tiré est déplacé le long de la ligne de visée proportionnellement à la vitesse $v_\parallel(z)$. Comme pour les quasars, la vitesse le long de la ligne de visée à $z = 0$ est donnée par

$$v_\parallel = \frac{v_{\rm X}{\rm X} + v_{\rm Y}{\rm Y} + v_{\rm Z}{\rm Z}}{\sqrt{{\rm X}^2 + {\rm Y}^2 + {\rm Z}^2}}\,. \tag{1.55}$$

1. https ://github.com/pyigm

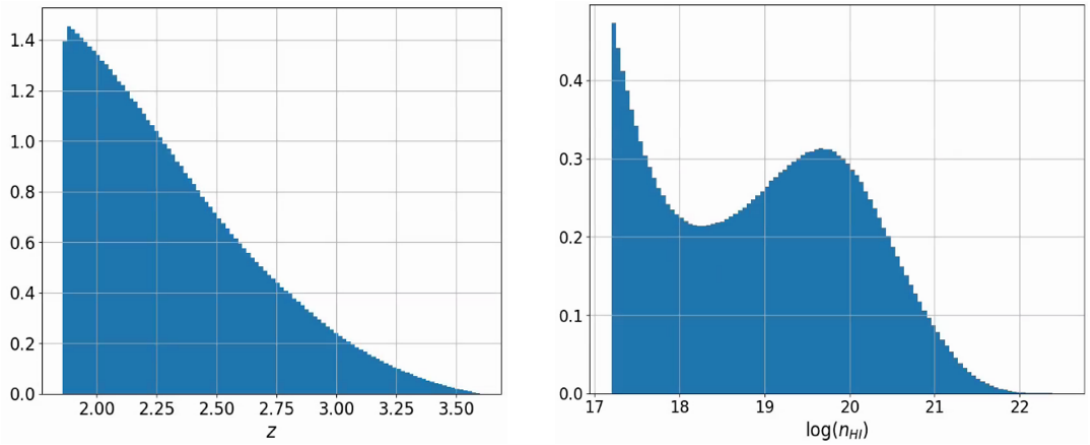

FIGURE 1.11 – Gauche : distribution normalisée en redshift des HCD. Droite : distribution normalisée de $\log(n_{\rm HI})$ des HCD. Ces distributions proviennent de la librairie `pyigm`.

où, cette fois ci, $v_{\rm X}$, $v_{\rm Y}$ et $v_{\rm Z}$ sont les vitesses le long de la ligne de visée provenant de l'interpolation des boîtes de vitesse (voir section 2.3). Ainsi, un HCD à un redshift z sera déplacé à un redshift

$$z \to z + \frac{(1+z)}{c} \frac{H(z)dG/dz}{[H(z)dG/dz]_{z=0}} v_{\parallel} \,. \tag{1.56}$$

Dans les mocks que nous décrivons ici, le profil d'absorption des HCD n'est pas ajouté dans les forêts. Nous produisons uniquement un catalogue qui regroupe tous les HCD tirés. Le profil d'absorption est ajouté au spectre de chaque quasar par le code `quickquasars`, qui utilise le catalogue de HCD que nous produisons.

3 Production des mocks

Comme expliqué au début de ce chapitre, l'objectif des mocks est de reproduire les données d'eBOSS et de DESI. Etant donné que le relevé d'eBOSS est contenu dans le relevé de DESI, nous simulons directement le relevé DESI. Ainsi, lorsque nous avons besoin de simuler le relevé d'eBOSS, nous retirons les quasars qui ne sont pas contenu dans ce relevé. Les mocks ont été produits au centre de calcul NERSC [1], avec la machine Cori. Sur cette machine, nous avons utilisé les nœuds "Haswell". Ces nœuds possèdent 128 Go de mémoire vive. De manière à créer nos boîtes de densité, via la transformation de Fourier à trois dimensions, il faut, pour chaque boîte, que l'intégralité de son contenu soit accessible depuis un même endroit. Nous pourrions distribuer la mémoire et effectuer la transformation de Fourier sur plusieurs nœuds à l'aide de la librairie MPI. Cependant nous n'avions pas l'expertise nécessaire. Nous effectuons donc les transformations de Fourier sur un seul nœud. Nous profitons néanmoins des 32 cœurs par nœud pour paralléliser notre code. Chaque cœur possède 2 hyper-threads, ce qui permet de gérer 64 tâches simultanément sur un même nœud. **Lors de la construction des boîtes (voir section 2.1), nous avons besoin de stocker en mémoire la boîte dans l'espace réel, la boîte dans l'espace k, et la boîte contenant la norme de k, cette dernière étant utilisée pour construire les boîtes de vitesses et de gradients de vitesse (équations 1.9 et 1.12).** Etant donné la mémoire disponible sur chaque nœud, les boîtes ne doivent pas dépasser 42 Go. Cette mémoire disponible et la taille des voxels choisie (2,19 h^{-1} Mpc) ne permettent pas de générer l'entièreté du relevé DESI avec une seule boîte. Nous choisissons donc de découper le relevé en

1. National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

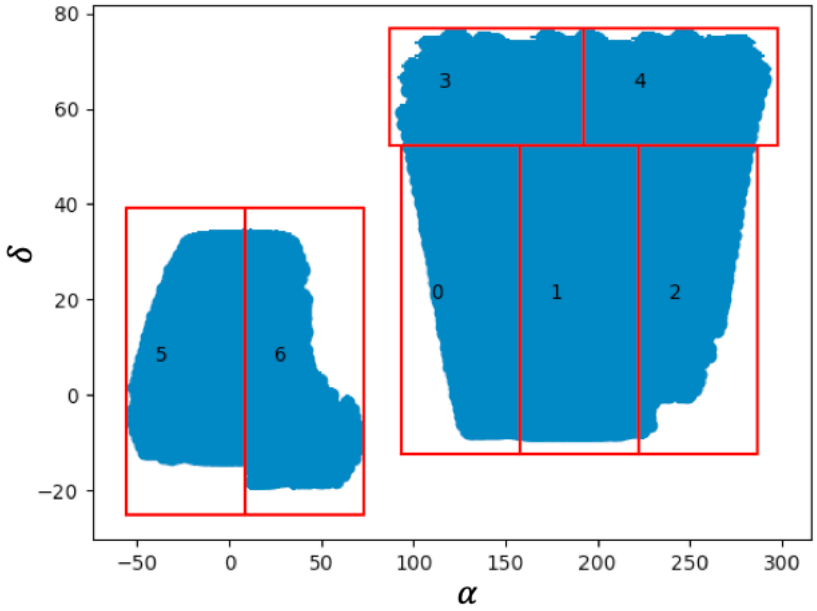

FIGURE 1.12 – Découpage du relevé de DESI en 7 chunks.

sept chunks indépendants, chacun étant généré par une boîte de $2560 \times 2560 \times 1536$ voxels. Chacune des boîtes représente un volume de 37,5 Go. A la fin de la production, ces sept chunks sont assemblés pour reconstruire le relevé de DESI. Le découpage du relevé en sept chunks est montré sur la figure 1.12. Dans les lignes qui suivent, nous détaillons les différents éléments du code. Le schéma 1.13 résume la situation. Le première module, `interpolate_pk.py` permet de calculer puis d'interpoler les quatres spectres de puissance $P_{\mathrm{QSO},i}$ et $P_{matière}$ sur la grille $2560 \times 2560 \times 1536$ dans l'espace k. Le code est lancé séparément sur 16 morceaux de la boîte, puis le code `merge_pk.py` permet de rassembler des 16 morceaux des spectres de puissance interpolés et de les sauver au format FITS (Flexible Image Transport System). Cette étape est effectuée une seule fois, car les spectres de puissance sont communs à toutes les réalisations. Le module suivant est `make_boxes.py`. Ce code lit les spectres de puissance interpolés puis construit les trois boîtes de densité relatives aux quasars, la boîte de densité relative au Lyα, les trois boîtes de vitesse et les six boîtes de gradient de vitesse, décrites dans la section 2.1. Au total, 13 transformations de Fourier inverses sont effectuées. Ce code est lancé sept fois, afin de produire les boîtes pour les sept chunks. Une fois toutes les boîtes produites, les quasars sont tirés (section 2.2) grâce au code `draw_qso.py`. Afin d'accélerer la production des mocks, les boîtes sont partagées en 512 tranches selon l'axe x. Ces tranches, de taille $5 \times 2560 \times 1536$, sont traitées séparément. Ce code est donc tourné en parallèle 512 fois, sur 16 nœuds $\times$ 32 threads. Une fois les quasars tirés, les lignes de visée sont interpolées (section 2.3) avec le code `make_spectra.py`. De la même manière que le code précédent, il tourne en parallèle 512 fois, sur 16 nœuds $\times$ 32 threads. Chaque instance du code interpole les densités, vitesses et gradients de vitesse le long de chaque morceau de ligne de visée présente dans la tranche traitée. Une fois toutes les tranches traités, les morceaux de ligne de visée sont mis bout à bout grâce au code `merge_spectra.py`. Encore une fois, le code tourne en parallèle 512 fois : chaque instance du code traite toutes les lignes de visée correspondant aux quasars situés dans une même tranche. Le code lit alors tous les morceaux de spectre relatifs à ces quasars, puis les assemble. Lorsque les lignes de visée sont toutes reconstruites, la formule FGPA est appliquée afin d'obtenir le champ de transmission F pour chaque ligne de visée. La dernière étape consiste à regrouper le résultat de tous les chunks. Le module `merge_qso.py` permet de lire tous les quasars tirés dans chaque tranche de chaque chunk et de créer un catalogue global appelé `master.fits`. Une fois le catalogue construit, le code `dla_saclay.py` tire les HCD le long de chaque ligne de visée. Le code

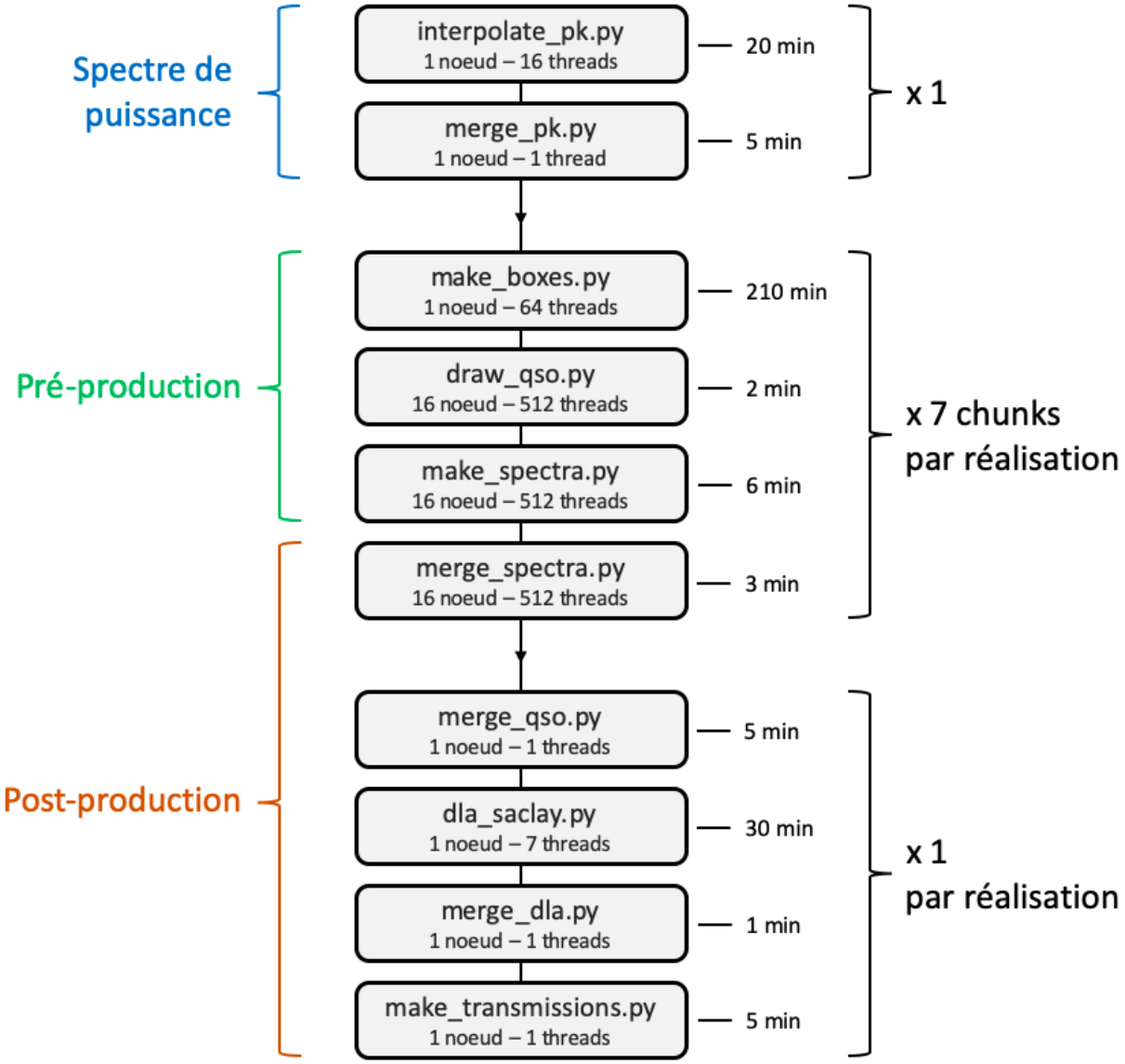

FIGURE 1.13 – Schéma illustratif du fonctionnement des mocks. Le premier bloc (bleu) génère le spectre de puissance nécessaire à la création des boîtes. Puis, pour chaque chunk de chaque réalisation, la pré-production (vert) génère les boîtes, tire les quasars, et interpole les lignes de visée. Enfin, la post-production (orange) rassemble les morceaux de lignes de visée et construit les forêts Lyα pour chaque chunk. Puis, elle regroupe le résultat de chaque chunk, construit les catalogues finaux de quasars et HCD et produit les fichiers transmissions pour chaque réalisation. Chaque case indique le nom du code, ainsi que les conditions dans lesquelles il est tourné. Les temps d'exécution indiqués sont approximatifs.

est tourné sur les sept chunks en parallèle. Puis le module `merge_dla.py`, comme pour les quasars, permet de regrouper tous les HCD tirés et de construire le catalogue global `master_DLA.fits`. Enfin le code `make_transmissions.py` permet de mettre les fichiers contenant les forêts au bon format : Les forêts sont regroupées par pixel HEALPix (Gorski et al. 2004) dans des fichiers FITS, puis ces fichiers FITS sont regroupés par 100 selon leur pixel HEALPix :

`N/PIX/transmission-nside-PIX.fits.gz` ,

où `nside` = 16 est la résolution du schéma HEALPix utilisé, `PIX` donne le numéro du pixel HEALPix, et `N` est le résultat de la division euclidienne de `PIX` par 100. Cette dernière étape est effectuée sur un nœud, `make_transmissions.py` étant lancé en parallèle sur 64 sous-échantillons des pixels HEALPix.

Comme expliqué à la fin de la section **??**, le calcul de l'auto corrélation des traceurs objets, tels les quasars et les HCD, nécessite l'estimateur de Landy-Szalay (voir calcul section **??**). Afin d'utiliser cette estimateur, nous avons besoin de catalogues de quasars et de HCD qui suivent une distribution aléatoire. Les distributions issues des vrais catalogues sont ensuite comparées aux distributions issues des catalogues aléatoires. Pour constuire ces catalogues aléatoires de quasars et HCD, nous utilisons les codes `draw_qso.py` et `dla_saclay.py`. Une option nous permet de tirer les objets sans tenir compte de la densité dans chaque voxel ou pixel. Les quasars sont donc tirés uniformément dans la boîte. Les HCD sont tirés uniformément le long de chaque ligne de visée, avec $z_{\rm HCD} < z_{\rm QSO}$ où $z_{\rm QSO}$ est le redshift du quasar hôte. Dans les deux cas, nous tirons plus de quasars et de HCD pour les distributions aléatoires que pour les vrais catalogues, afin que la statistique des fonctions de corrélation soit limitée uniquement par les vrais catalogues. A la fin, les catalogues aléatoires de quasars et de HCD contiennent respectivement environ 10 et 3 fois plus d'objets que les vrais catalogues.

Pour les analyses Lyα d'eBOSS et de DESI, nous avons décidé de produire 100 réalisations des mocks, afin d'avoir suffisamment de statistique pour étudier finement les potentielles systématiques. Nous avons organisé la production de ces 100 réalisations en deux étapes. Premièrement, nous avons effectué la *pré-production*. Cette étape consiste à créer les boîtes contenant les différents GRF, puis à tirer les quasars et enfin interpoler la densité, les vitesses et gradients de vitesse, le long de chaque ligne de visée. Ceci correspond aux codes `make_boxes.py`, `draw_qso.py` et `make_spectra.ppy`. Cette étape est la plus coûteuse en temps de calcul : environ 43 heures CPU sur un nœud de Cori pour produire les sept chunks d'une réalisation. L'ensemble des fichiers temporaires propres à une réalisation pré-produite représente $\sim 7 \times 550\,$Go sur disque. **La majorité de cet espace disque est pris par les différentes boîtes. Afin de réduire cet espace disque, la seule boîte que nous sauvons est la boîte contenant δ_k. A partir de cette boîte δ_k, nous pouvons reconstruire toutes les autres boîtes, puis reconstruire le relevé de quasars et les lignes de visée. Ceci nous permet de vérifier la production des réalisations. Ainsi, pour chaque réalisation pré-produite, nous stockons les lignes de visée interpolées et la boîte δ_k, ce qui représente $\sim 340\,$Go sur disque.**

Une certain nombre de problèmes informatiques, liés au centre de calcul NERSC, ont ralenti la phase de pré-production. Le principal problème venait du temps de lecture et d'écriture, qui par moment pouvait être multiplié par un facteur cent. Plus de trentes secondes étaient parfois nécessaires pour accéder à un simple fichier. Ce problème a été identifié comme venant du transfert des *meta data* sur le centre de calcul. Le code le plus affecté est `make_boxes.py`, car il écrit énormément de fichiers différents, correspondant aux différents slices des boîtes, et destinés à être lus par les codes `draw_qso.py` et `make_spectra.py`. Pour palier ce problème, nous avons décidé d'essayer de faire tourner les codes sur les nœuds de Cori appelés *Burst Buffer*. Ces nœuds possèdent des disques SSD (Solid-State drive), ce qui permet une lecture et une écriture très rapide. Une fois les codes exécutés, les données produites sont déplacées sur les disques durs habituels. Les nœuds Burst Buffer ont permis d'accélérer l'exécution du code `make_boxes.py` par un facteur ~ 2, passant d'environ deux heures à une heure seulement. Une vintaine de réalisation ont été produites en utilisant ces nœuds. Cependant,

ils sont devenus instables au cours de la production. D'autre part, les problèmes liés aux transferts des meta data avaient été stabilisés entre temps. Nous sommes donc retournés à l'utilisation des nœuds classiques de Cori pour finir la production. Pour les 40 dernières réalisations, le temps d'exécution de `make_boxes.py` variait entre trois et quatres heures.

Le temps pris par les six mois nécessaires à la production [1] **nous a permis de choisir sur quelle modélisation des données nous voulions ajuster les paramètres Lyα.** En effet, la modélisation des HCD dans les données est complexe et mal comprise. Les paramètres $b_{\rm HCD}$ et $\beta_{\rm HCD}$ sont très corrélés avec ceux du Lyα. Selon les modélisations, les paramètres Lyα obtenus grâce à l'ajustement des données ne sont pas les mêmes. L'étude de ces modélisations est présentée dans la section **??**. Après avoir bien défini les paramètres Lyα que nous visons, nous avons ajusté les paramètres de la formule FGPA afin d'obtenir les paramètres Lyα souhaités dans les mocks. Ceci est expliqué dans la section suivante. Une fois ces paramètres ajustés et la phase de pré-production terminée, nous avons mené la phase de *post-production*. Cette phase consiste à créer les spectres d'absorption à partir des densités interpolées le long des lignes de visée, puis à regrouper tous les fichiers de sortie afin de les mettre au format décrit précédemment. Cette étape est beaucoup plus rapide, elle prend l'équivalent d'environ 6 heures CPU sur un nœud de Cori par réalisation. Une fois la production complète effectuée, la place sur disque d'une réalisation, sans compter les fichiers temporaires, correspond à environ 15 Go.

4 Ajustement des paramètres

Comme expliqué dans la section 2.4, le champ d'absorption Lyα est construit à partir des boîtes δ_l, δ_s et $\eta_\parallel$ grâce à la formule FGPA. Contrairement aux quasars pour lesquels le biais est choisi, les paramètres physiques du champ d'absorption Lyα simulé dépendent des quatres paramètres $a(z)$, $b(z)$, $c(z)$ et $P_s(z)$ utilisés dans l'équation 1.40. Nous décrivons dans cette section comment ajuster ces paramètres (paramètres FGPA dans la suite) afin d'obtenir les bons $b_{\rm Ly\alpha}(z)$, $\beta_{\rm Ly\alpha}(z)$, $\overline{F}(z)$ et $P^{\rm 1D}(k,z)$ (paramètres Lyα dans la suite). Nous nous servons de la modélisation des données décrite dans le chapitre **??** comme référence pour les paramètres Lyα.

Afin d'ajuster ces paramètres FGPA, la méthode standard est de générer un mock avec un jeu de paramètres FGPA, calculer la fonction de corrélation du Lyα, ajuster cette fonction de corrélation et mesurer les paramètres Lyα. Puis une fois ces paramètres mesurés, itérer sur les paramètres FGPA afin de nous rapprocher des paramètres Lyα visés. Cependant cette méthode est très couteuse, car elle nécessite de générer des spectres et de produire la fonction de corrélation à chaque itération. Pour accélérer la procédure d'ajustement, nous tirons profit du fait que nos mocks possèdent une fonction de corrélation prédictible. **Ainsi, au lieu de générer des spectres et calculer la fonction de corrélation sur ces derniers à chaque itération, nous calulons la prédiction (détaillée en section 2.4) puis nous générons une pseudo-fonction de corrélation qui suit cette prédiction. Nous ajustons directement cette pseudo-fonction de corrélation afin de mesurer les paramètres Lyα correspondant au jeu de paramètres FGPA utilisé.** Cet ajustement est fait pour les cinq valeurs du redshift $z_1 = 1{,}8$; $z_2 = 2{,}2$; $z_3 = 2{,}6$; $z_4 = 3{,}0$ et $z_5 = 3{,}6$.

Dans les lignes qui suivent, nous expliquons comment, pour chaque valeur du redshift, nous choisissons le jeu de paramètres FGPA initial pour générer la prédiction à la première itération. **Premièrement, nous choisissons le paramètre c. Comme expliqué dans la section 2.4, le champ δ_s ne participe pas à la corrélation à trois dimension. De plus, la somme des champs δ_l et $\eta_\parallel$ permet de retrouver le champ δ dans l'espace des redshifts, via la formule de Kaiser (équation 1.13). Ce champ δ dans l'espace des redshift possède un biais de 1, par construction. Son paramètre RSD est donc $\beta = f/b = f$. De plus, à grand redshift, nous**

1. Dans de bonnes conditions, le temps de production de 100 réalisations est estimé à 4 à 6 semaines. #prov ça serait mieux dans le texte

pouvons considérer en bonne approximation que $f \sim 1$. Nous avons donc $\beta \sim 1$. Ainsi, lorsque nous considérons la somme des trois champs δ_l, δ_s et $\eta_\parallel$ dans l'équation 1.40, le terme c devant $\eta_\parallel$ donne, en première approximation, $\beta = c$. Pour chaque valeur du redshift, nous choisissons donc $c(z) = \beta_{\mathrm{Ly}\alpha}(z)$.

En ce qui concerne les paramètres a et b, ils sont ajustés afin de retrouver le bon $\overline{F}(z)$ et $P^{\mathrm{1D}}(k,z)$ (#prov décrire).

Une fois les paramètres a, b et c choisis, nous déterminons le spectre de puissance P_s à appliquer à $\delta_{k,s}$ afin d'obtenir le bon P^{1D}_{mock}. Nous avons besoin de P_s car il nous faut connaître σ_g, qui lui même requiert σ_s, pour chaque valeur du redshift afin de calculer la prédiction. La variance du champ $g = \delta_l + \delta_s + c(z)\eta_\parallel$ est donnée par

$$\sigma_g^2(z) = \langle \delta_l^2 \rangle + \langle \delta_s^2 \rangle + c^2(z)\langle \eta_\parallel^2 \rangle + c(z)\Big(\langle \delta_l \eta_\parallel \rangle - \langle \delta_l \rangle \langle \eta_\parallel \rangle\Big) \,. \tag{1.57}$$

Le terme entre parenthèses est la covariance des champs δ_l et $\eta_\parallel$. La variance σ_s du champ δ_s est reliée à son spectre de puissance P_s par l'équation 1.37. Dans le but de calculer σ_s puis la prédiction, nous commençons donc par constuire le spectre de puissance $P^{\mathrm{1D}}_{modèle}$ sur lequel le P^{1D}_{mock} des mocks sera ajusté. **Tout d'abord, les données provenant de (#prov On fit les données DR12 sans les oscillations du Si, quel papier ?) sont ajustées dans la gamme $0{,}2 < k < 2.0\boldsymbol{h}$ Mpc^{-1}. La fonction ajustée aux données est définie comme**

$$f(k) = \exp\left(-ak + b + \frac{c}{k+d}\right) \times f_{\mathrm{Si}}(k) \,, \tag{1.58}$$

où a, b, c et d sont quatres paramètres à ajuster. La fonction f_{Si} prend en compte les oscillations dues au silicium. Une fois la fonction f de l'équation 1.58 ajustée sur les données, nous utilisons comme modèle pour le spectre de puissance à une dimension la fonction f sans la contribution du silicium f_{Si}. Ceci nous permet de nous affranchir des oscillations dues au silicium dans notre modèle. La forme choisie pour f permet d'avoir $\log\left(P^{\mathrm{1D}}_{modèle}\right)$ linéaire à grand k, en accord avec ce qui est mesuré dans les simulations hydrodynamiques (Arinyo-i-Prats et al. 2015). Pour construire δ_s, nous avons besoin de calculer P_s jusqu'à k_{Ny}. Le résultat de l'ajustement sur les données est donc extrapolé de $k = 2{,}0$ jusqu'à $k = 20\,h$ Mpc^{-1}. Pour les k plus petits que $0{,}2\,h$ Mpc^{-1}, nous utilisons le modèle décrit dans Arinyo-i-Prats et al. (2015) et ajusté sur des simulations hydrodynamiques. Il est défini comme

$$P(k) = b^2_{\mathrm{Ly}\alpha}(1 + \beta_{\mathrm{Ly}\alpha}\mu_k^2)^2 P_{\mathrm{L}}(k) D(k,\mu) \,, \tag{1.59}$$

où P_{L} est le spectre de puissance linéaire, donné par Camb, et $D(k,\mu)$ représente les déviations par rapport à la théorie linéaire. Le terme $D(k,\mu)$ tend donc vers 1 à petit k. La forme choisie dans Arinyo-i-Prats et al. (2015) diffère de celle utilisée dans McDonald (2003). Cette nouvelle forme permet d'obtenir le bon comportement à petit k. Aussi, elle nécessite l'ajustement de moins de paramètres. $D(k,\mu)$ est donc défini comme

$$D(k,\mu) = \exp\left[\Big(q_1\Delta^2(k) + q_2\Delta^4(k)\Big)\left(1 - \left(\frac{k}{k_v}\right)^{a_v}\mu^{b_v}\right) - \left(\frac{k}{k_p}\right)^2\right] \,, \tag{1.60}$$

avec

$$\Delta^2(k) = \frac{1}{2\pi^2}k^3 P_L(k) \,. \tag{1.61}$$

Les termes $q_1\Delta^2(k)$ et $q_2\Delta^4(k)$ représente l'augmentation de puissance aux petites échelles due aux non-linéarités. L'ajustement que nous utilisons force $q_2 = 0$. Les autres paramètres ajustés sont donnés dans la section "Planck" de la table 7 de Arinyo-i-Prats et al. (2015). Puis, pour chaque valeur du

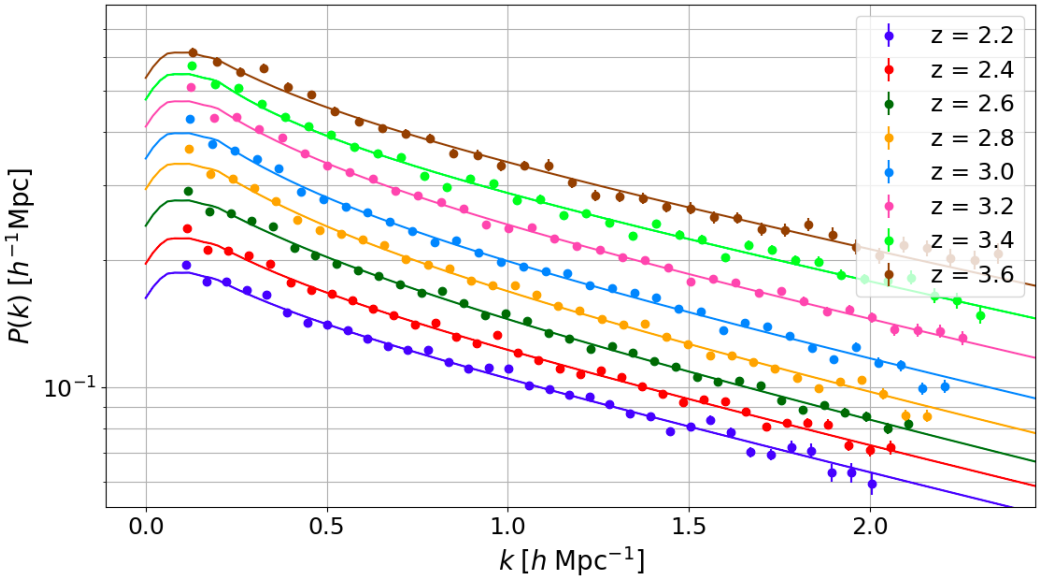

FIGURE 1.14 – Modèle du spectre de puissance à une dimension, pour différentes valeurs du redshift z, utilisé dans les mocks. Le modèle, donné par les lignes continues, est ajusté sur les données (**prov**), représentées par les points. Ce modèle est décrit dans le texte.

redshift, la forme ajustée aux données et définie dans l'équation 1.58 est prolongée de $k = 0{,}2\,h$ Mpc^{-1} jusqu'à $k = 0$ par le modèle donné dans l'équation 1.60. Pour ce faire, nous calculons le modèle à $z = 2{,}4$, puis nous le multiplions par une constante, dépendant de la valeur du redshift, de façon à ce que le prolongement en $k = 0{,}2\,h$ Mpc^{-1} soit continu. **La figure 1.14 montre le modèle ainsi construit, avec les données de #prov.** Pour le bin $z = 1{,}8$, aucune donnée Lyα n'est disponible. Nous extrapolons donc le modèle à $z = 2{,}2$. Nous considérons que la forme du modèle reste la même. Nous considérons aussi que l'évolution en redshift de $z = 2{,}4$ à $z = 2{,}2$ est la même jusqu'à $z = 1{,}8$. Le modèle à $z = 1{,}8$ est donc donné par

$$P^{\mathrm{1D}}_{modèle}(k, z = 1{,}8) = P^{\mathrm{1D}}_{modèle}(k, z = 2{,}2) \left(\frac{P^{\mathrm{1D}}_{modèle}(k, z = 2{,}2)}{P^{\mathrm{1D}}_{modèle}(k, z = 2{,}4)} \right)^2 . \tag{1.62}$$

Une fois le spectre de puissance modèle défini, nous estimons le spectre de puissance à une dimension P^{1D}_l du champ interpolé $\delta_l + c\eta_\parallel$. Nous partons du spectre de puissance fournit par Camb. Nous multiplions ce spectre de puissance par le terme de kaiser $(1 + c\mu_k^2)^2$ puis par le terme W^2 représentant l'effet du lissage gaussien (équation 1.30). Enfin, nous calculons P^{1D}_l grâce à l'équation 1.33, en nous restreignant aux $k < k_{max}$. Une première estimation du spectre de puissance à une dimension P^0_s à appliquer à $\delta_{k,s}$ afin d'obtenir le bon P^{1D}_{mock} est donnée par

$$P^0_s(k) = P^{\mathrm{1D}}_{modèle}(k) - P^{\mathrm{1D}}_l(k) . \tag{1.63}$$

Le spectre de puissance P^0_s ainsi construit nous permet d'obtenir un P^{1D}_{mock} convenable. **Cependant, il reste des différences avec le spectre de puissance modèle.**Afin de corriger ces différences, nous ajustons itérativement la forme de P_s. A chaque itération n, nous commençons par générer δ_s selon P^n_s. Puis nous calculons le P^{1D}_{mock}, correspondant à ce P^n_s. Le P^{n+1}_s de l'itération suivante est alors donné par

$$P^{n+1}_s(k) = P^n_s(k) \frac{P^{\mathrm{1D}}_{modèle}(k)}{P^{\mathrm{1D}}_{mock}(k)} . \tag{1.64}$$

Quelques itérations sont suffisantes pour obtenir un P_s qui donne un P^{1D}_{mock} en accord avec les données. Nous en effectuons dix pour chaque valeur du redshift.

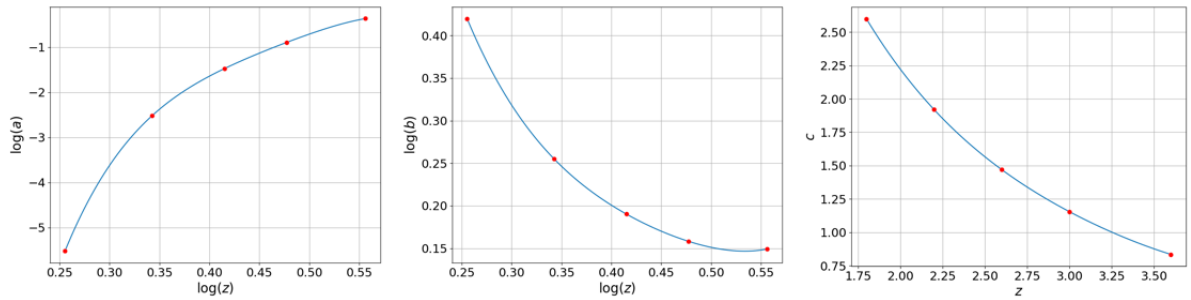

FIGURE 1.15 – Résultat de la procédure d'ajustement des paramètres FGPA.

A ce stade, nous disposons des informations nécessaires pour calculer la prédiction. Afin de mesurer les paramètres Lyα prédits, nous générons une pseudo-fonction de corrélation qui suit la prédiction calculée avec le jeu de paramètres. Nous ajustons ensuite cette pseudo-fonction de corrélation avec le code picca, comme nous le ferions avec les mocks. Dans le chapitre ??, nous verrons que les paramètres Lyα sont corrélés entre eux. Ainsi, afin de comparer au mieux la prédiction avec les données, plutôt que de mesurer les paramètres $b_{\mathrm{Ly}\alpha}$ et $\beta_{\mathrm{Ly}\alpha}$, nous mesurons les paramètres $b_{\mathrm{eff,Ly}\alpha}$ et $\beta_{\mathrm{Ly}\alpha}$. Le biais effectif $b_{\mathrm{eff,Ly}\alpha}$ est relié au biais et au paramètre RSD du Lyα par

$$b_{\mathrm{eff,Ly}\alpha} = b_{\mathrm{Ly}\alpha}\sqrt{1+\frac{2}{3}\beta_{\mathrm{Ly}\alpha}+\frac{1}{5}\beta^2_{\mathrm{Ly}\alpha}}\,. \tag{1.65}$$

Il est sensible à l'amplitude de la fonction de corrélation, et est moins corrélé avec $\beta_{\mathrm{Ly}\alpha}$ que l'est $b_{\eta,\mathrm{Ly}\alpha}$ ou $b_{\mathrm{Ly}\alpha}$. Enfin, nous mesurons aussi $\overline{F}$ dans la prédiction, et nous comparons sa valeur à ce qui est mesuré dans les données (#prov Papier de White, mais lequel ?).

L'estimation initiale que nous faisons des paramètres b et c donnent des valeurs de $\beta_{\mathrm{Ly}\alpha}$ et $\overline{F}$ proches de ce qui est mesuré dans les données. Cependant, la valeur de a est surestimée : les paramètres a et b sont estimés à partir des mesures de P^{1D} et $\overline{F}$ dans les données. Comme nous le verrons dans la section ??, la présence de HCD dans les données a pour effet d'augmenter le biais effectif mesuré. Ainsi, l'estimation du paramètre a, est faite avec un P^{1D} qui possède un biais trop grand. Ceci résulte en une surestimation du paramètre a, et donc un biais effectif trop grand dans les mocks. Nous itérons alors sur les paramètres a, b et c : nous diminuons a, et si besoin modifions légèrement b et c afin d'affiner les valeurs de $\beta_{\mathrm{Ly}\alpha}$ et $\overline{F}$ des mocks. Puis, nous recalculons le nouveau P^{10}_s afin de générer la nouvelle prédiction. Nous ajustons de nouveau la prédiction avec `picca` et comparons les valeurs de $b_{\mathrm{eff,Ly}\alpha}$, $\beta_{\mathrm{Ly}\alpha}$ et $\overline{F}$ mesurées aux données. Ces itérations sont faites jusqu'à obtenir des valeurs en accord avec les données.

Une fois que cette procédure itérative a convergé pour les cinq valeurs du redshift, nous créons des fonctions continues des paramètres $a(z)$, $b(z)$ et $c(z)$ sur $z \in [1{,}8\,;3{,}6]$. Pour $a(z)$ et $b(z)$, nous calculons les polynômes de degré quatre de $\log z$ qui passent par les cinq points $\log a(z)$ et $\log b(z)$. Ceci est fait dans le but d'éviter les valeurs négatives pour $a(z)$, ainsi que les points d'inflection. Pour le paramètre $c(z)$, nous calculons le polynôme de z qui passe par les cinq points $c(z)$. La figure 1.15 présente les fonctions continues ainsi obtenues. En ce qui concerne les cinq spectres de puissance $P_s(k)$ construits précédemment pour chaque valeur du redshift, nous créons une grille (k,z) couvrant $z \in [1{,}8\,;3{,}6]$ et $k \in [0\,;20]h$ Mpc^{-1}. Puis, $P_s(k,z)$ est obtenu via une interpolation cubique sur cette grille. Du fait des très faibles valeurs de P_s à grand k et de l'interpolation cubique utilisée, certaines valeurs de l'interpolation sont négatives. Pour chaque redshift de la grille, nous forçons tous les pixels de l'interpolation de P_s à zéro pour tous les $k > k_0$, où k_0 est le premier k pour lequel l'interpolation est nulle. La figure 1.16 montre

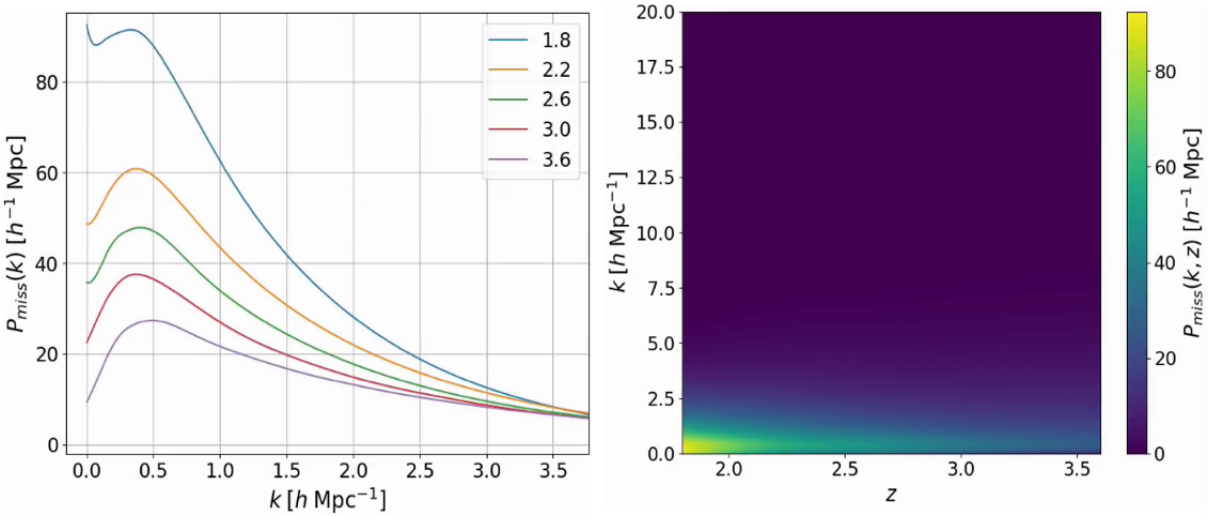

FIGURE 1.16 – Gauche : les cinqs spectres de puissance $P_s(k)$ ajustés pour chaque valeur du redshift. Droite : interpolation cubique de ces cinqs spectres de puissance sur une grille (k, z).

l'interpolation de $P_s(k)$ pour différentes valeurs de z.

Le résultat de la procédure d'ajustement décrite précédemment a été utilisé pour produire 100 réalisations indépendantes des mocks. Les figures **??** présentent les paramètres Lyα obtenus avec la prédiction (couleur), sur le stack des 100 rea (couleur) et mesuré dans les données (couleur). #prov montrer biais(z), beta(z), <F>(z) et P1D(z).

5 Expansion des mocks

Les mocks décrits précédemment produisent un relevé de quasars avec, pour chaque quasar, une forêt d'absorption F variant entre 0 et 1. Afin de reproduire complètement les données et pouvoir simuler la chaîne d'analyse, nous devons ajouter un continuum à ces forêts, puis les inclure dans des spectres simulés. Le code utilisé pour créer ces spectres synthétiques est le code `quickquasars`. Il fait parti du package `desisim` [1] et est décrit dans **prov**

1. https ://github.com/desihub/desisim

Bibliographie

Agrawal, Aniket et al. (2017). « Generating Log-normal Mock Catalog of Galaxies in Redshift Space ». In : doi : 10.1088/1475-7516/2017/10/003. arXiv : 1706.09195.

Arinyo-i-Prats, Andreu et al. (2015). « The Non-Linear Power Spectrum of the Lyman Alpha Forest ». In : doi : 10.1088/1475-7516/2015/12/017. arXiv : 1506.04519.

Clerkin, L. et al. (2016). « Testing the lognormality of the galaxy and weak lensing convergence distributions from Dark Energy Survey maps ». In : doi : 10.1093/mnras/stw2106. arXiv : 1605.02036.

Coles, Peter et Bernard Jones (1991). « A lognormal model for the cosmological mass distribution ». In : *Monthly Notices of the Royal Astronomical Society* 248, p. 1–13. issn : 0035-8711.

Dodelson, Scott (2003). *Modern Cosmology*. Academic Press. doi : 10.1023/B:FOOP.0000019699.88151.ed.

Farr, James et al. (2019). « LyaCoLoRe: Synthetic Datasets for Current and Future Lyman-alpha Forest BAO Surveys ». In : doi : 10.1088/1475-7516/2020/03/068. arXiv : 1912.02763.

Font-Ribera, Andreu, Patrick Mcdonald et Jordi Miralda-Escudé (2012). « Generating mock data sets for large-scale Lyman- α forest correlation measurements Generating mock data sets for large-scale Lyman-α forest correlation measurements ». In : *Journal of Cosmology and Astroparticle Physics*. doi : 10.1088/1475-7516/2012/01/001.

Font-Ribera, Andreu et Jordi Miralda-Escudé (2012). « The Effect of High Column Density Systems on the Measurement of the Lyman alpha Forest Correlation Function ». In : doi : 10.1088/1475-7516/2012/07/028. arXiv : 1205.2018.

Gorski, K. M. et al. (2004). « HEALPix – a Framework for High Resolution Discretization, and Fast Analysis of Data Distributed on the Sphere ». In : doi : 10.1086/427976. arXiv : 0409513 [astro-ph].

Hamilton, A. J. S. (1992). « Measuring Omega and the real correlation function from the redshift correlation function ». In : *The Astrophysical Journal* 385, p. L5. issn : 0004-637X. doi : 10.1086/186264.

Heitmann, Katrin et al. (2019). « The Outer Rim Simulation: A Path to Many-Core Supercomputers ». In : doi : 10.3847/1538-4365/ab4da1. arXiv : 1904.11970.

Le Goff, J. M. et al. (2011). « Simulations of BAO reconstruction with a quasar Lyman-alpha survey ». In : *Astronomy \& Astrophysics* 534, A135. issn : 0004-6361. doi : 10.1051/0004-6361/201117736. arXiv : 1107.4233.

Lewis, Antony, Anthony Challinor et Anthony Lasenby (1999). « Efficient Computation of CMB anisotropies in closed FRW models ». In : doi : 10.1086/309179. arXiv : 9911177 [astro-ph].

McDonald, Patrick (2003). « Toward a Measurement of the Cosmological Geometry at z ˜ 2: Predicting Lyα Forest Correlation in Three Dimensions and the Potential of Future Data Sets ». In : *The Astrophysical Journal* 585.1, p. 34–51. issn : 0004-637X. doi : 10.1086/345945.

Pieri, M. M. et al. (2016). « WEAVE-QSO: A Massive Intergalactic Medium Survey for the William Herschel Telescope ». In : arXiv : 1611.09388.