Statistiques sur générations WCRG et comparaisons avec d'autres modèles

Jean-Eric Campagne, Etienne Lempereur

1er septembre 2023

Table des matières

1. Introduction

Dans cette note est exposé une série de statistiques calculées sur des images générées par WCRG (Guth et al. 2023) en les comparant aux mêmes statistiques calculées non seulement avec les images ayant servies au training du modèle, mais aussi pour un jeu de données avec d'autres modèles comme un DGAN et un modèle micro-locale.

Brièvement, les données utilisées sont les suivantes:

WL-1 : la motivation première était de comparer les images de Weak Lensing [1] issues d'un modèle Deep GAN de la référence (Mustafa et al. 2019) avec un modèle WCRG entrainé à partir des mêmes données d'entrainement que le réseau génératif. Donc, pour le training les auteurs nous ont fourni 100,000 images 128x128 pixels, ainsi qu'un lot d'images générées par leur réseau CosmoGAN. Ainsi, nous n'avons donc pas ré-entrainé un DGAN à cette occasion. Nous avons extrait 5,000 images [2] pour l'entrainement d'un modèle WCRG (noté par la suite WCRG-WL1). Nous détaillons un peu plus le traitement dans la section **??**.

WL-2 & ϕ_4 : A la suite, de la comparaison entre le modèle WCRG-WL1 et le DGAN, nous nous sommes interrogés sur ce que les modèles WCRG entrainés sur des données WL mentionnées dans le papier (Guth et al. 2023). Nous avons donc repris ces images d'entrainement et synthétisées pour calculer les statistiques. Idem nous avons utilisé les images d'entrainement et synthétisées par WCRG ϕ_4 ($\beta = \{0.5, 0.68, 0.76\}$ pour calculer les dites statistiques. Cela est détaillé dans la section **??**

Les statistiques utilisées sont principalement issues du code python *scattering_transform* développé par Siho Cheng [3] et utilisé entres autres dans les articles (Cheng & Ménard 2021; Cheng et al. 2023). Cela concerne, le power, le bi et le tri spectra et aussi les coefficients de corrélation C_{00} et C_{01}. Concernant les données de WL, la statistique du comptage de pics a également été utilisée.

Concernant la modélisation WCRG, le code python utilisé est celui du repository https://github.com/Elempereur/WCRG/ mentionné dans la publication afférente.

1. La référence utilise des résultats de la simulation N-corps GadGet2.
2. C'est la statistique utilisée dans la référence (Guth et al. 2023)
3. https://github.com/SihaoCheng/scattering_transform commit Fri Jun 2 11:14:53

2. Modélisation WCRG sur les données WL-1

2.1 Phase d'apprentissage

Avant l'entrainement du modèle WCRG, un traitement des données que l'on peut qualifier de "gaussianisation" a été effectué. En effet, sur la figure 1, on peut remarquer que la distribution des valeurs des pixels est mono-mode et peut être transformée en distribution normale par la méthode Cox-Box suivie d'une standardisation classique [4]. Le résultat de la transformation globale est illustré sur la figure 2. Cette transformation globale (inversible) est appliquée sur tous les lots de d'images (entrainement et génération) du DGAN.

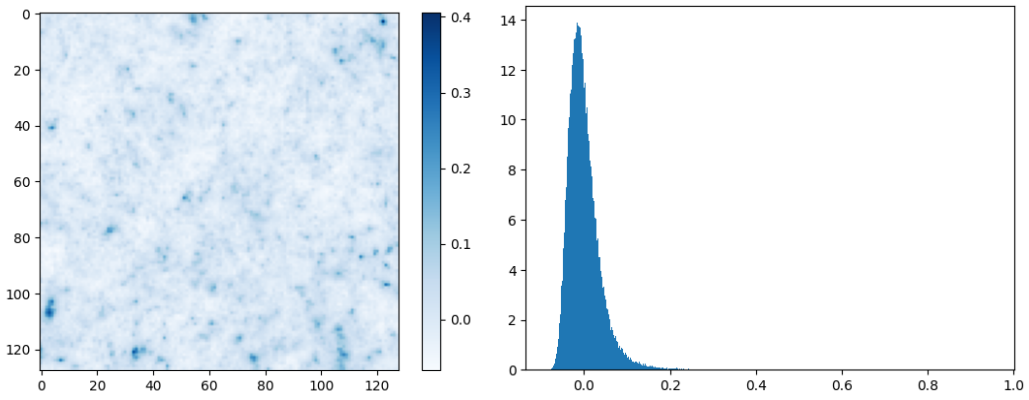

FIGURE 1 – Exemple d'une image du lot WL-1 (gauche) et la distribution afférente des valeurs de pixels (droite).

Concernant l'optimisation du modèle WCRG [5], la démarche est très proche de celle documenté sur le repository du code, en particulier le notebook *Learning_Models.ipynb* moyennant quelques adaptations décrites ci-après.

4. Si x est la valeur d'un pixel original alors primo par l'optimisation Cox-Box on trouve $\lambda^* \approx -0.22$, et $x' = (x^{\lambda^*} - 1)/\lambda^*$ suit une loi gaussienne, puis la standardisation classique permet d'obtenir une distribution normale $x' \sim \mathcal{N}(0, 1)$.

5. Notons que l'optimisation du modèle pourrait sans doute se passer de l'opération de gaussianisation préalable, mais l'idée était que cela pourrait aider *a priori* de ne pas avoir à traiter des queues de distributions trop asymétriques.

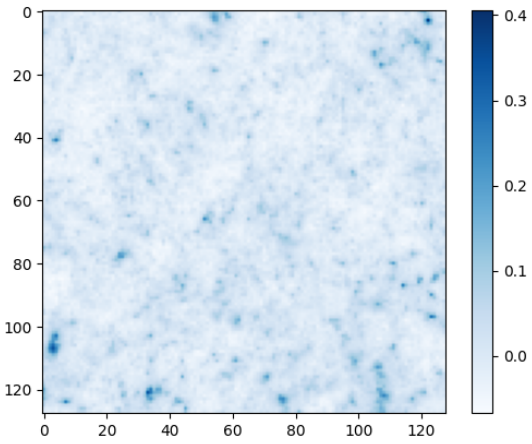

FIGURE 2 – Distribution des valeurs de pixels de la figure 1 après application d'une transformation Cox-Box de paramètre $\lambda \approx -0.22$ et d'une standardisation classique.

- Les ondelettes utilisées sont de la familles Debauchies d'ordre 4 avec un padding périodique.
- Pour l'ansatz $E(x_J)$ ($J = 7$, $L = 1$) a été utilisée la fonction

```
def ANSATZ_NoCondi(L,centers,sigma,shifts,shifts_sym = False):
    """Non conditionnal ansatz for direct estimation of energy, a scalar
    potential (sigmoids) + a quadratic potential

    Parameters:
    L (int): system size = L*L
    centers (tensor): position of the centers of the sigmoids
    sigma (tensor) : width of the sigmoids
    shifts (list of tuples) : spatial shifts for the quadratic potential,
    carefull (0,0) is already taken into account, do not add here
    shifts_sym (Bool) : if True, the shifts are not symetrized
    """
```

  avec 20 sigmoïdes régulièrement espacées entre le min et le max de la distribution des pixels $\approx [-100, 100]$, et par ailleurs shifts $= ()$). L'optimisation SGD est menée après avoir normalisé le hessien.
- A toutes les échelles $L > 1$, nous avons utilisé la fonction

```
def ANSATZ_Wavelet(W,L,centers,sigma,mode,shifts,shifts_sym = False):
""" Conditionnal ansatz for conditonal Energy \bar E(\bar x_j\vert x{j})
```

```
    estimation with a wavelet transform,with a scalar potential (sigmoids) + a
    quadratic potential

    Parameters:
    W (Wavelet) : Wavelet to perfom fast wavelet transform
    L (int): system size ( of x_{j-1}) = L*L
    centers (tensor): position of the centers of the sigmoids
    sigma (tensor) : width of the sigmoids
    shifts (list of tuples) : spatial shifts for the quadratic potential,
    carefull (0,0) is already taken into account, do not add here
    shifts_sym (Bool) : if True, the shifts are not symetrized
"""
```

Tout comme pour E_J, l'optimisation SGD est menée après avoir normalisé le hessien.

- A l'échelle $L = 2$, 30 sigmoïdes régulièrement espacées dans l'intervalle $[-75, 75]$ sont utilisées, tandis que extend = 1.0. Les autres paramètres sont mode = 'All' et shifts = ().
- A l'échelle $L = 4$, 30 sigmoïdes sont également utilisées mais cette fois réparties uniformément selon les quantiles avec $q_{min} = 10^{-5}$, $q_{max} = 1.0$ et extend = 1.0. De plus mode = 'All' tandis que shifts = ((1,0),(0,1),(1,1)) conjointement à shifts_sym = True.
- A l'échelle $L = 8$, on utilise 40 sigmoïdes réparties uniformément selon les quantiles avec $q_{min} = 10^{-5}$, $q_{max} = 1.0$ et extend = 1.0. Par ailleurs outre mode = 'All', nous avons utilisé la fonction suivante pour définir les shifts tels que shifts =shift_modif(L//4).

```
def shift_modif(n):
    shifts =[]
    for i in range(-n,n):
        for j in range(-n,n):
            if i==0 and j==0:
                pass
            else:
                shifts.append((i,j))
    return shifts
```

- A l'échelle $L = 16$, se sont 40 sigmoïdes uniformément réparties sur l'intervalle $[-30, 30]$ que l'on utilise avec extend = 1.0. Par ailleurs mode = 'All' et shifts

=shift_modif(L//4).

- A l'échelle $L = 32$, on utilise comme précédemment 40 sigmoïdes uniformément réparties sur l'intervalle $[-20, 20]$ (extend = 1.0). Les paramètres extend, mode et shifts sont les mêmes valeurs pour les deux premiers et définition pour le dernier que pour l'échelle précédente.
- A l'échelle $L = 64$, on utilise 30 sigmoïdes réparties uniformément selon les quantiles avec $q_{min} = 10^{-5}$, $q_{max} = 1.0$ et extend = 1.0. De plus si mode = 'All' comme précédemment, par contre shifts = shift_modif(L//8).
- Enfin à l'échelle $L = 128$ (taille des images d'entrainement), on utilise comme 30 sigmoïdes uniformément réparties sur l'intervalle $[-7.0, 5.5]$ (extend = 0.25), et si mode = 'All' comme précédemment, par contre shifts = shift_modif(L//16).
- Les paramètres qui sont particulièrement délicats affectant la synthèse des images aux différentes échelles, sont ceux qui définissent les centres des sigmoïdes (l'intervalle $[min, max]$ pour linspace_centers, ou $[q_{min}, q_{max}]$ pour quantile_centers. La méthode shift_modif a été utilisée in fine mais ne change pas vraiment les résultats par rapport à l'usage de shift_quad_Sym utilisée de prime abord. Les optimisations SGD sont souvent reprises pour un peu tuner les paramètres tels que le nombre d'époques num_epochs et le learning rate lr. Parfois deux étapes d'optimisation ont été pratiquées, notamment avec une valeur de lr plus petite pour la seconde étape.

Une fois les paramètres "optimisés" (par ex. les sigmoïdes), les différents potentiels appris $\bar{v}_j(x_{j-1})$ sont présentés sur la figure 3.

2.2 Phase de synthèse

Une fois le modèle appris, on peut regarder à chaque échelle la qualité des synthèses. En fait, à dire vrai, les paramètres qui définissent les sigmoïdes à toutes les échelles ont été ajustés à la main largement pour que les histogrammes de contrôles montre une bonne adéquation entre images d'origines et images de synthèse. La technique de sampling est en tout point identique à celle mise en œuvre pour l'article. Ce qui peut changer concerne le nombre de steps et la taille de chaque step. *Même si la suite pourrait paraître fastidieuse et répétitive, dans cette note une certaine exhaustivité s'impose néanmoins surtout que les codes n'ont pu être tournés sur notebooks partageables.*

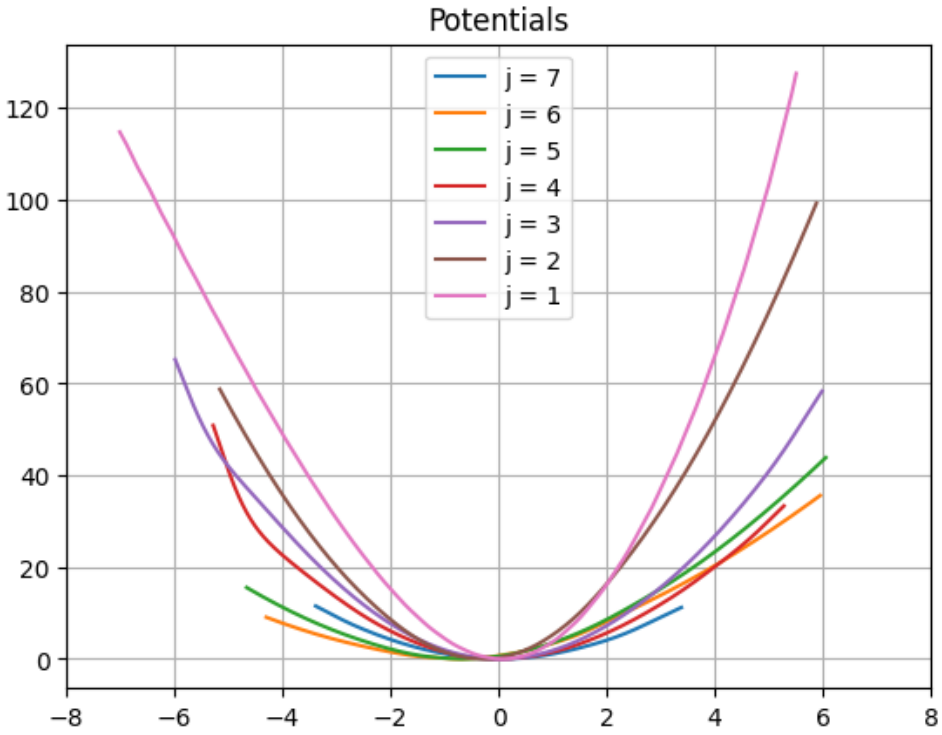

FIGURE 3 – Potentiels scalaires équivalents $\bar{v}_j$ appris à chaque échelle j pour les images de WL-1.

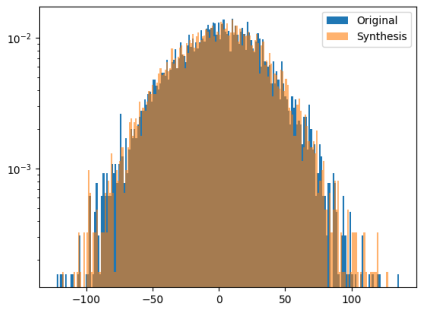

FIGURE 4 – Comparaison des valeurs de pixels à l'échelle $L = 1$.

A l'échelle $L = 1$, la comparaison des distributions de valeurs de pixels entre données d'entrainement et synthétisées est montrée sur la figure 4.

Aux échelles suivantes, outre la comparaison des distributions des valeurs de pixels des images originales et synthétisées, on peut également questionner la distribution des cartes de détails hautes fréquences (horizontales/verticales/diagonales) obtenues par décomposition en ondelettes.

Les résultats aux échelles $L \in [1, 128]$ sont présentés sur les figures de 5 à 11. Pour des raisons tenant à la taille mémoire des GPUs utilisés, le nombre de cartes générées jusqu'à l'échelle 16 incluse est de 5000, tandis que pour les échelles supérieures on ne peut disposer que de 500 cartes.

A l'échelle $L = 128$, on dispose de cartes synthétisées "finales" qui si tout se passe au mieux doivent ressembler comme deux gouttes d'eau aux cartes utilisées pour l'entrainement (ou cartes "originales"). La figure 12 donne quelques exemples de cartes originales et synthétisées. Il est indéniable que le modèle a appris quelque chose. Maintenant, dans l'article (Guth et al. 2023), il est montré non seulement des cartes synthétisées, la distribution des valeurs de pixels [6] mais aussi la comparaison du spectre de puissance. Ce dernier point a motivé l'usage de statistiques afin de caractériser les distributions des pixels et de comparer ces statistiques obtenues sur les cartes originales et les synthétisées. Ceci est présenté dans la section suivante.

6. nb. le modèle de l'article a été optimisé directement sur les cartes de WL-2 sans transformation préalable donc on ne peut comparer les modèles. Cela sera sans doute à revoir ultérieurement.

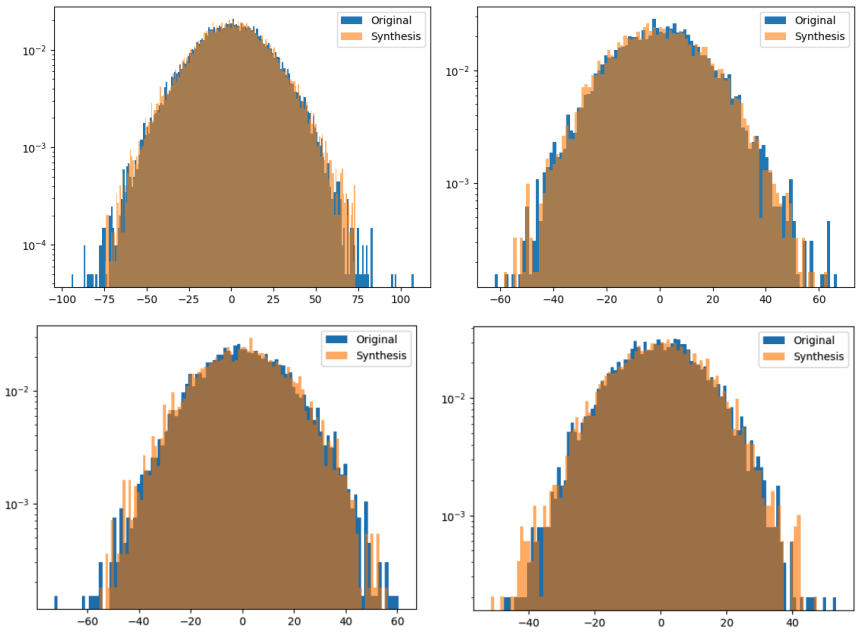

FIGURE 5 – Comparaison à l'échelle $L = 2$ entre cartes de type "originale/entrainement" et de type "synthètisée": distribution des valeurs de pixels de cartes "totales" (haut-gauche), de détails "H" (haut-droite), de détails "V" (bas-gauche) et de détails "D" (bas-droite).

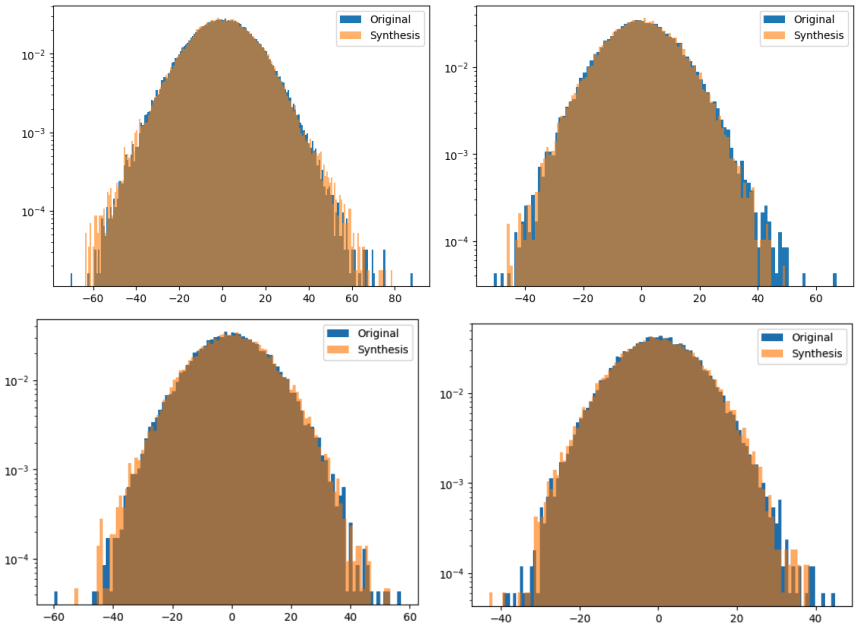

FIGURE 6 – Echelle $L = 4$: légende identique à la figure 5.

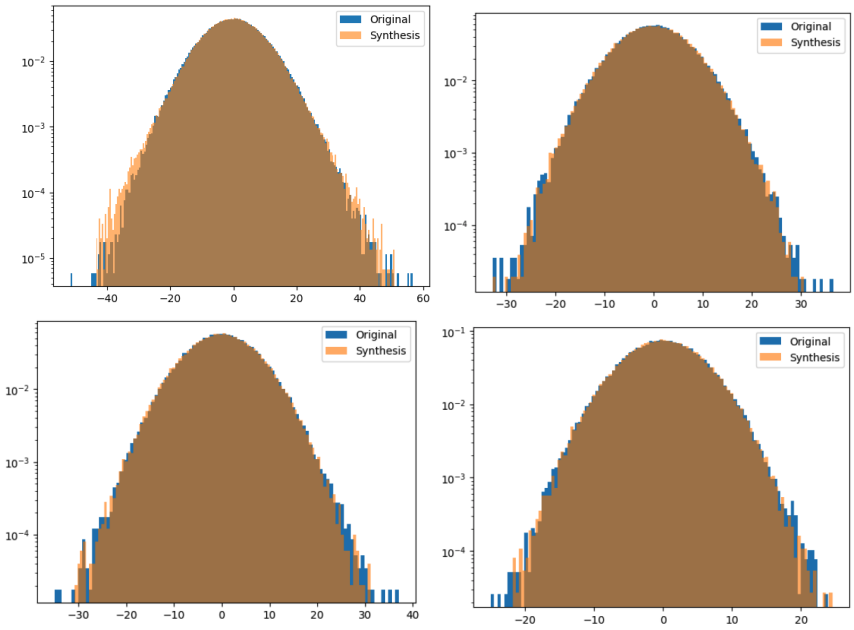

FIGURE 7 – Echelle $L = 8$: légende identique à la figure 5.

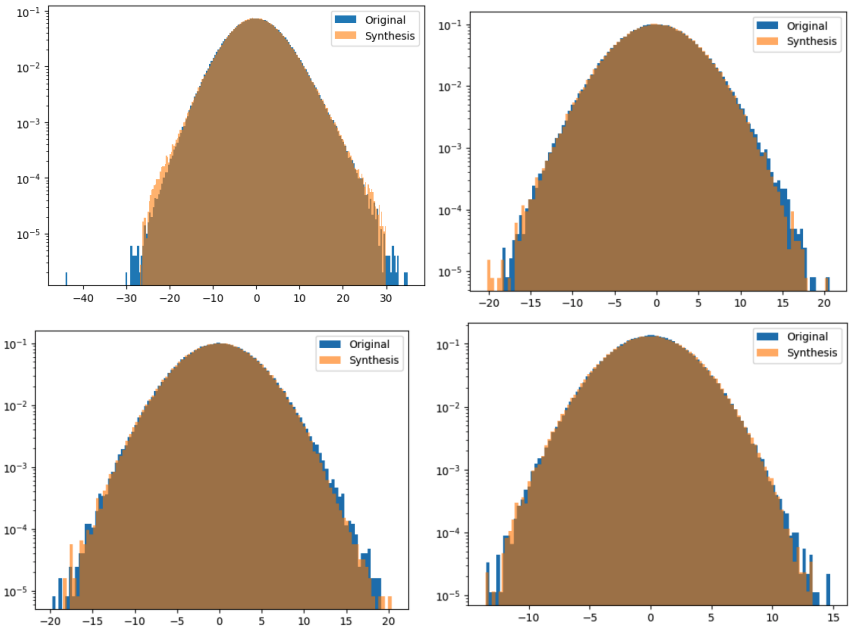

FIGURE 8 – Echelle $L = 16$: légende identique à la figure 5.

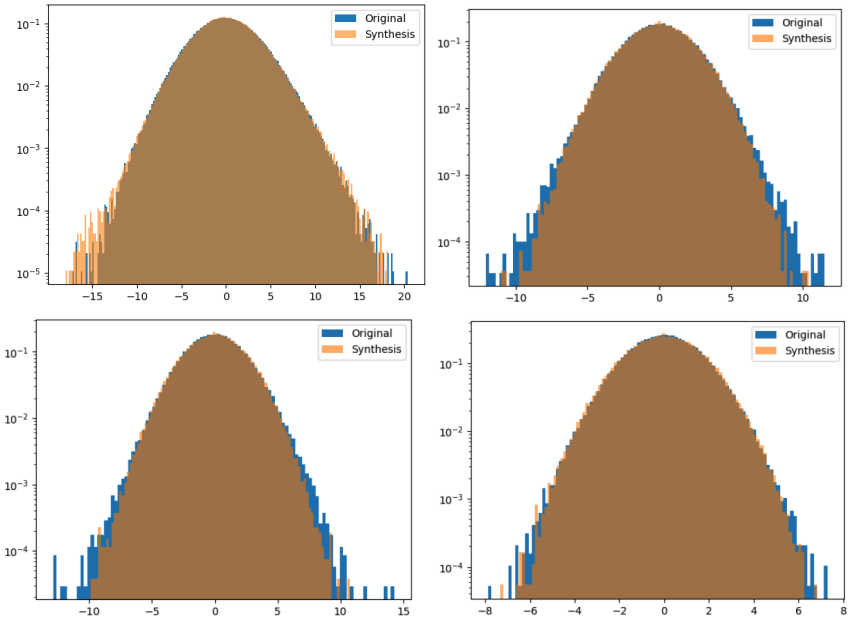

FIGURE 9 – Echelle $L = 32$: légende identique à la figure 5.

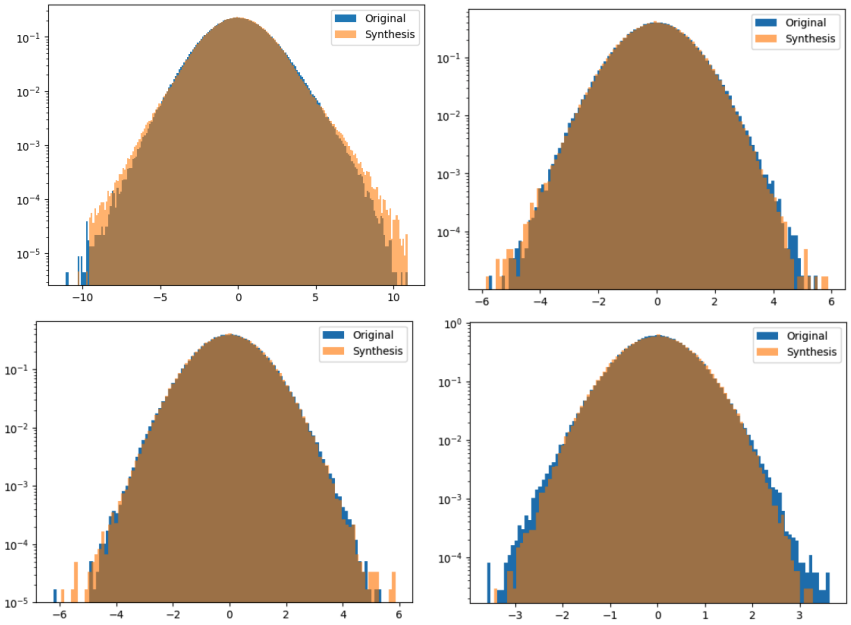

FIGURE 10 – Echelle $L = 64$: légende identique à la figure 5.

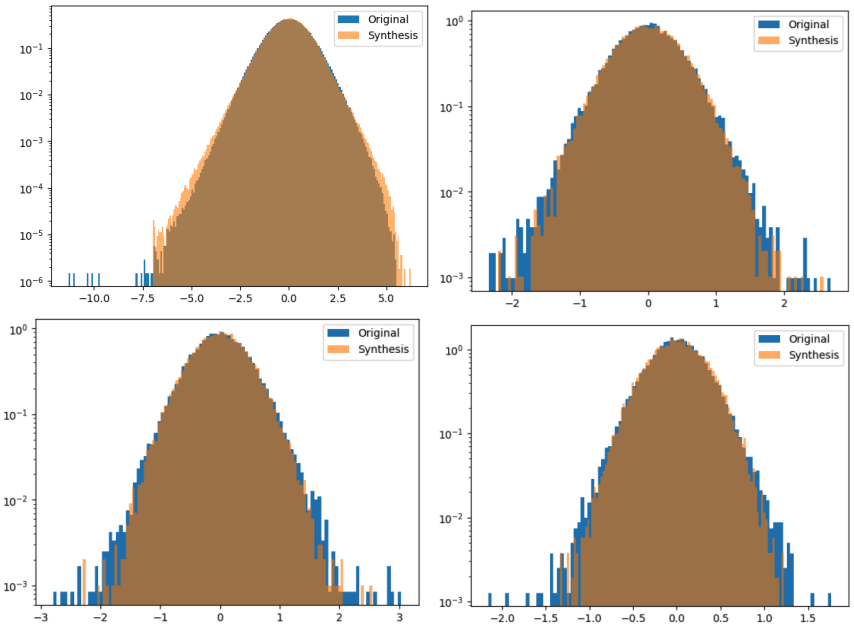

FIGURE 11 – Echelle $L = 128$ (la plus grande possible): légende identique à la figure 5.

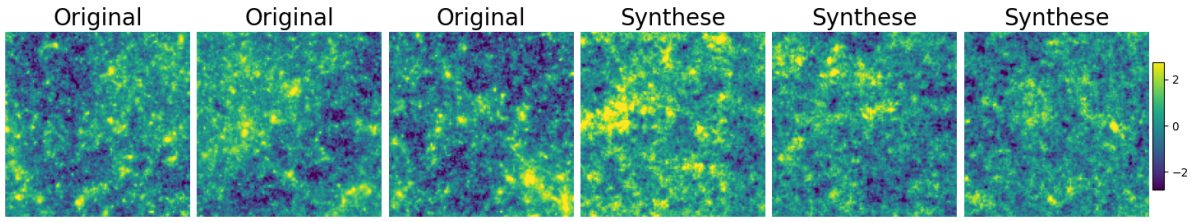

FIGURE 12 – Exemples de cartes WL-1 utilisées pour l'entrainement et synthétisées une fois le modèles WCRG optimisé.

3. Réduction d'information par l'usage de statistiques

Un certain nombre de statistiques ont été utilisées afin d'aller au-delà de l'aspect visuel jugeant de la qualité de la synthèse des cartes de WL-1. Comme mentionné dans l'introduction, en premier lieu il a été utilisé le code développé par Siho Cheng afin de calculer:

- les classiques spectre de puissance, *bi*-spectre et *tri*-spectre;
- le coefficient de scattering $S1_{iso}$, ainsi que les coefficients composites s_{21} (sparsity) et s_{22} (shape) définis dans la référence (Cheng & Ménard 2021);
- mais aussi les versions *isotropiques* des corrélations de scattering définies selon [7]

$$C_{01} = \langle (I * \psi_2)(|I * \psi_1| * \psi_2)^* \rangle / factor \quad (1)$$

$$C_{11} = \langle (|I * \psi_1| * \psi_3)(|I * \psi_2| * \psi_3)^* \rangle / factor \quad (2)$$

avec I la carte de champ, ψ_i des ondelettes de Morlet à différentes échelles et orientations et où le choix de normalisation est défini par

$$factor = L2(I * \psi_1) \times L2(I * \psi_2) \quad (3)$$

Les définitions de ces facteurs sont décrites dans l'article (Cheng et al. 2023): C_{01} et C_{11} correspondent respectivement à $\tilde{S}_3$ et $\tilde{S}_4$, et la version "iso" opère une réduction sur les paires d'angles.

En pratique afin de fixer un peu les idées, l'obtention des différents spectres et coef-

7. $\langle x \rangle = \underset{u}{\text{Ave}}\,(x(u))$.

ficients s'effectue schématiquement selon les étapes suivantes (si "maps" correspond à un lot de cartes):

```
bins_pw = 30
Nmaps,M,N = maps.shape
J = int(np.log2(min(M,N))) - 1
pw, kr  = scattering.get_power_spectrum(maps,
              k_range=np.logspace(0,np.log10(M/2*1.4),
              bins_pw+1))
bi_calc = scattering.Bispectrum_Calculator(M,N,
              k_range=np.logspace(0,np.log10(M/2*1.4), J-1))
tri_calc = scattering.Trispectrum_Calculator(M,N,
              k_range=np.logspace(0,np.log10(M/2*1.4), J-1))

st_calc = scattering.Scattering2d(M, N, J=6, L=4)

coeffs = st_calc.scattering_coef(maps)
s1  = np.log10(coeffs['S1_iso'])
s21 = np.log10(coeffs['s21'])
s22 = np.log10(coeffs['s22'])

cov_coef = st_calc.scattering_cov(maps)
select_and_index = get_scattering_index(J, L, normalization='P00', C11_criteria='j2>=
    j1')
c01 = cov_coef['C01_iso'][:,select_and_index['select_2_iso']]
c11 = cov_coef['C11_iso'][:,select_and_index['select_3_iso']]
```

De plus, afin de satisfaire des contraintes de taille de mémoire, les spectres/coefficients ont été calculés sur des lots de cartes dont on en a tiré des moyennes et écarts-types (barres d'erreur dans les histogrammes).

Dans le cas des cartes de WL, on a également utilisé le comptage de "pics" au dessus d'un seuil. Il s'agit d'une statistique d'ordre supérieur à la recherche des non-gaussianité qui s'est répandue récemment (voir la référence Lanzieri et al. (2023) et les références incluses). Pour se faire la méthode détaillée dans l'article de D. Lanizieri et al. a été utilisée [8] avec à la base la fonction find_peaks2d_tf du package lenspack [9] qui cherche la localisation des pics ainsi que leurs "hauteurs", puis un lissage par un noyau gaussien. Le

8. https://github.com/LSSTDESC/DifferentiableHOS.git
9. https://github.com/CosmoStat/lenspack

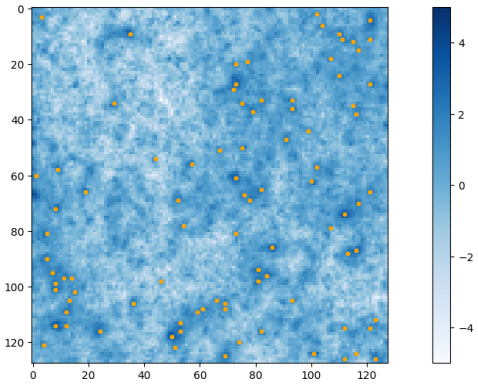

FIGURE 13 – Exemples de la recherche de pics au dessus d'un seuil ici pris à 2.

résultat est donc un histogramme de la distribution smoothée des hauteurs de pics d'une carte (ou d'un lot de cartes). Un exemple de recherche de pics pour un seuil de 2 est donné sur la figure 13.

4. Résultats concernant les données WL-1

Une fois le modèle WCRG obtenu selon la méthode détaillée à la section 2.1, on peut calculer les statistiques décrites dans la section précédente non seulement sur des cartes générées (Sec. 2.2) mais aussi sur des cartes ayant servi à l'apprentissage du modèle. Les résultats sont présentés sur la figure 14 sous les appellations "wcrg" (rouge) et "train" (bleu). Il est assez remarquable que la quasi totalité des distributions issues de la génération de cartes par le modèle WCRG soient bien, voire très bien reproduite. Donc, il n'est pas étonnant que les différences que nous pointons par la suite n'aient pas été mise en évidence dans l'article (Guth et al. 2023).

Si on se penche sur les distributions qui dénotent, on constate que:

- en premier lieu ce qui questionne concerne la distribution C_{01}^{iso} qui manifestement a un problème, alors que la distribution C_{11}^{iso} est bien reproduite;
- dans une moindre mesure on constate aussi que le bi-spectre (s_{21}) est moins bien reproduit que le tri-spectre (s_{22}).

C'est deux points dont l'origine peut être la même, a motiver les investigations suivantes.

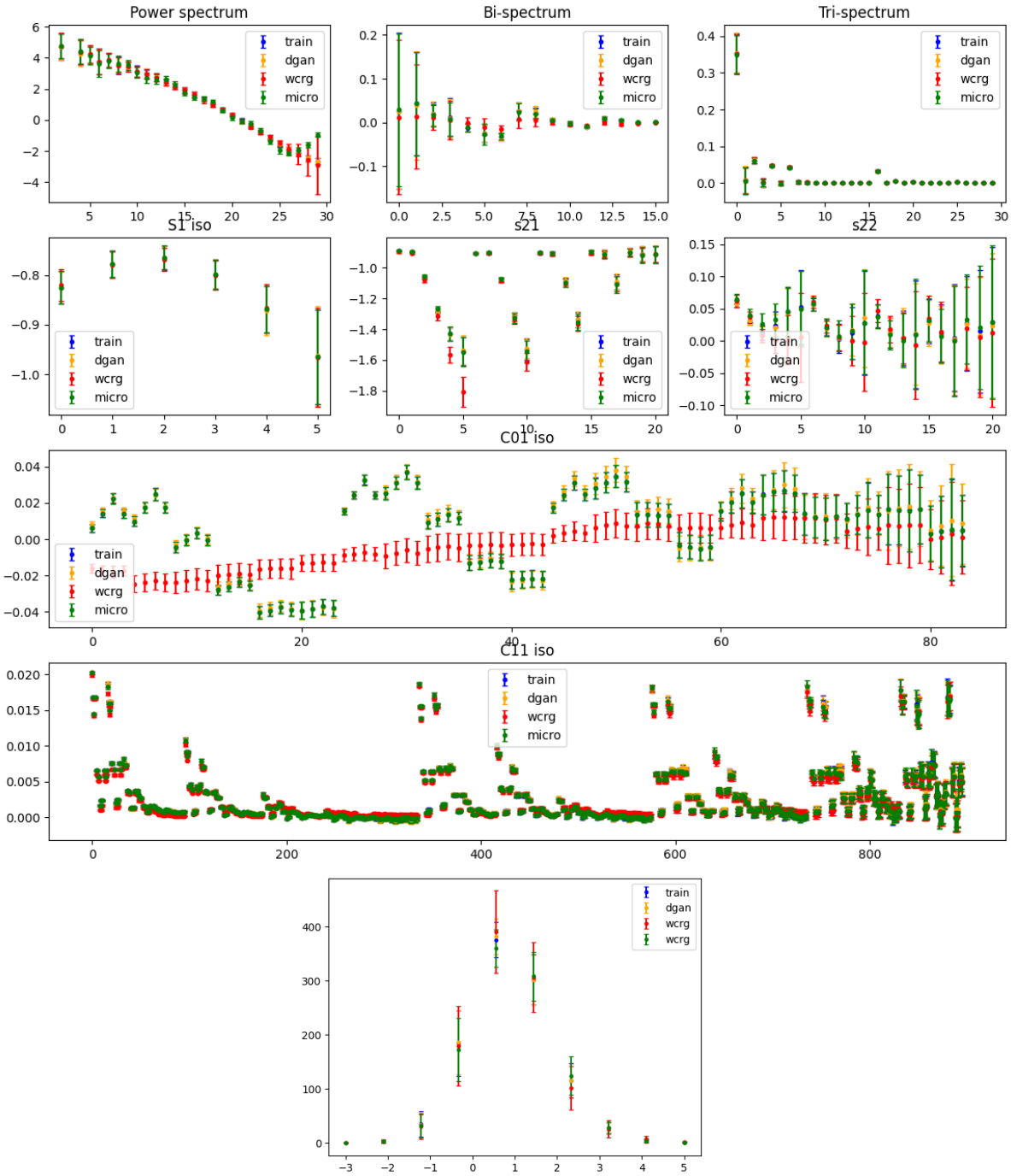

FIGURE 14 – Histogrammes des statistiques détaillées à la section 3. Il est a noté qu'excepté le comptage de pics, les distributions sont calculées avec 500 cartes.

Notons que si l'on prend les cartes issues de la génération du réseau CosmoGAN mentionné en introduction, il y a un parfait accord de toutes les distributions "dgan" (orange) avec les distribution d'entrainement ("train"). En ce sens, le GAN fait mieux sur ce critère [10].

Nous avons également procédé à la synthèses de cartes en utilisant une modélisation micro-canonique (Cheng et al. 2023) à l'aide du code de Siho Cheng selon typiquement

```
st_calc.scattering_cov(...)['for_synthesis']
```

qui utilise les paramètres: le rapport $\langle I \rangle / \sigma(I)$, $L2(I * \psi_i)$, S_1, ainsi que les parties réelles et imaginaires des coefficients de corrélation C_{01} et C_{01} Ce type de modélisation/synthèse notée "micro"(vert) sur la Fig. 14 reproduit très bien la quasi totalité des distributions exceptée [11] le spectre de puissance à grande valeur de k même si on note que les fluctuations des valeurs sont bien plus réduites que pour les autres modèles pourtant calculées avec le même nombre de cartes. Ceci, concernant les coefficients C_{01} l'accord est parfait, mais on s'en doutait puisqu'ils rentrent dans la liste des paramètres ayant servi pour définir la métrique à minimiser afin de produire le modèle micro-canonique.

Dans les sections suivantes, nous allons utiliser les cartes générées par les modèles WCRG ainsi que celles d'entrainement de type WL-2 et ϕ_4 décrites dans l'introduction, afin de savoir si ces modèles performent mieux ou pas.

5. Modèles WCRG et données WL-2 et ϕ_4

La question qui vient est de savoir pour quelles raisons la distribution C_{01}^{iso} (et aussi bi-spectre et s_{21}) est nettement moins bien reproduite par le modèle WCRG optimisé comme décrit à la section 2.1. Pour se faire, nous avons utilisé directement (c'est-à-dire sans nouveau entrainement) les modèles et les données de l'article (Guth et al. 2023).

Concernant les données WL-2, on dispose pour le calcul des statistiques de 500 cartes d'entrainement ("train") et de 209 cartes générées ("wcrg"). Une note technique: nous

10. Rappel: à l'origine de l'étude était la comparaison "dgan", "wcrg", et dans ce contexte le nombre d'images ayant servit à l'entrainement du modère WCRG est de 5,000 images alors que le DGAN en a utilisé 200,000 (Mustafa et al. 2019)

11. Si l'on ne porte que les distributions "train" et "micro", la remontée du spectre de puissance "micro" est bien visible.

avons procéder à un recalage de la distribution des valeurs de pixels des cartes générées afin que la moyenne soit égale à celle des cartes d'entrainement, mais nous n'avons pas opéré de gaussianisation (cf. Sec. 2.1). Les résultats sont présentés sur la figure 15. Par rapport à la figure 14 utilisant un modèle optimisé sur les données WL-1 (gaussianisée), on constate

- primo que la distribution des C_{01}^{iso} calculées avec les cartes générées ("wcrg") suit les ondulations de la distribution calculée sur les données d'entrainement ("train"). Mais il y a un petit désaccord néanmoins mais bien plus petit qu'en même.
- les petits désaccords sur les distribution bi-spectre et s_{21} sont de même ampleur
- on note également petit désaccord sur la distribution des C_{11}^{iso}.

Ce premier point nous renseigne sur le fait qu'un modèle WCRG est capable de générer des cartes avec des coefficients C_{01}^{iso} convenables en tous les cas bien plus que ceux de la figure 14.

Concernant les données ϕ_4, on dispose de 3 lots obtenus avec les valeurs de β différentes ($\{0.5,\ 0.68,\ 0.76\}$) qui influencent la dynamique sous-jacente. Tout comme pour les cartes WL-2, on a procédé à l'accord des moyennes des distributions des valeurs de pixles entre les cartes générées et les cartes d'entrainement. Les différents résultats sont exposés sur les figures 16, 17, et 18.

Ce que l'on constate c'est que:

- les distributions obtenues avec les données $\beta = 0.50$ sont remarquablement concordantes; et notons que les coefficients C_{01}^{iso} même sur les données d'entrainement sont quasi-nuls.
- les désaccords entre les distributions avec les données $\beta = 0.76$ sont assez semblables à ceux observés sur les données WL.
- la figure des distributions des coefficients C_{01}^{iso} avec les données $\beta = 0.76$ est particulière: les coefficients calculés avec les données générées sont quasi-nuls, or on constate une structuration de la distribution calculée avec les données d'entrainement. Notons que contrairement au cas WL la distribution du bi-spectre semble bien reproduite (aux erreurs statistiques près), et il en va de même avec la distribution s_{21}.

Ainsi, on dispose de plusieurs cas de figures et à part 1 cas (ϕ_4, $\beta = 0.50$) tous les autres exhibent des divergences concernant principalement les coefficients C_{01}^{iso}. Même si

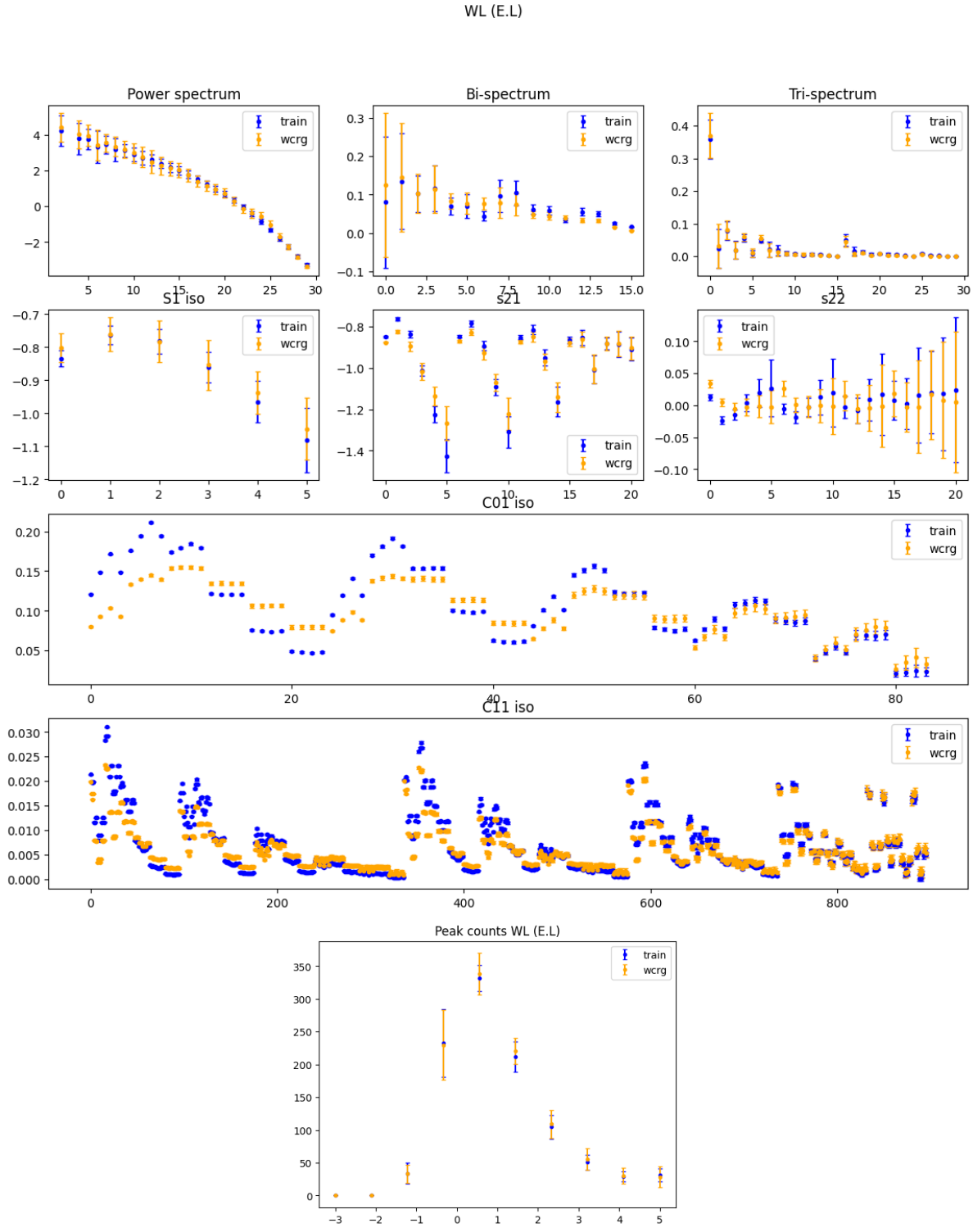

FIGURE 15 – Statistiques de la figure 14 mais concernant les données WL-2.

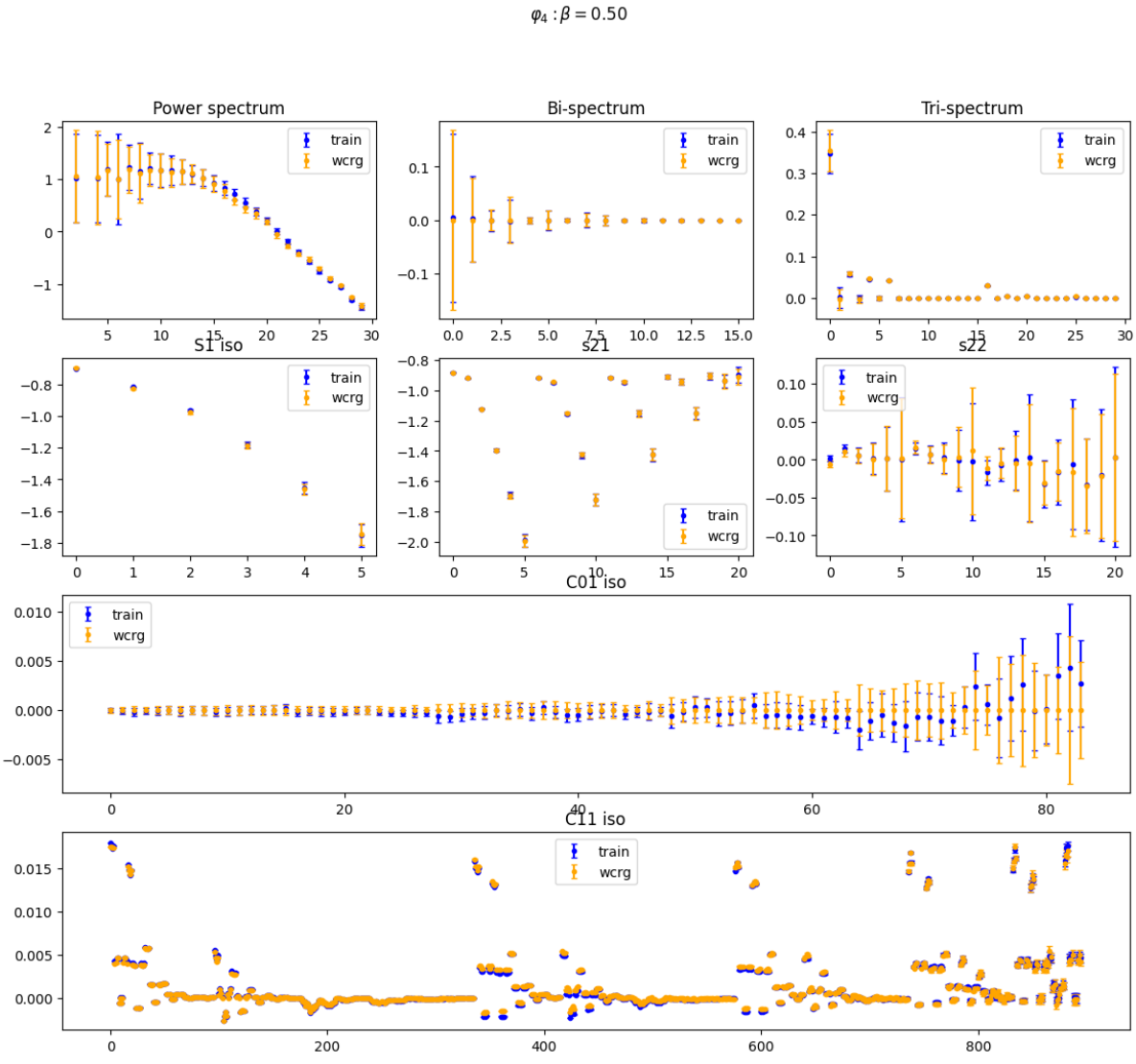

FIGURE 16 – Statistiques de la figure 14 mais concernant les données ϕ_4 ($\beta = 0.50$). Notons que l'on dispose de 500 cartes "train" et de 400 cartes "wcrg".

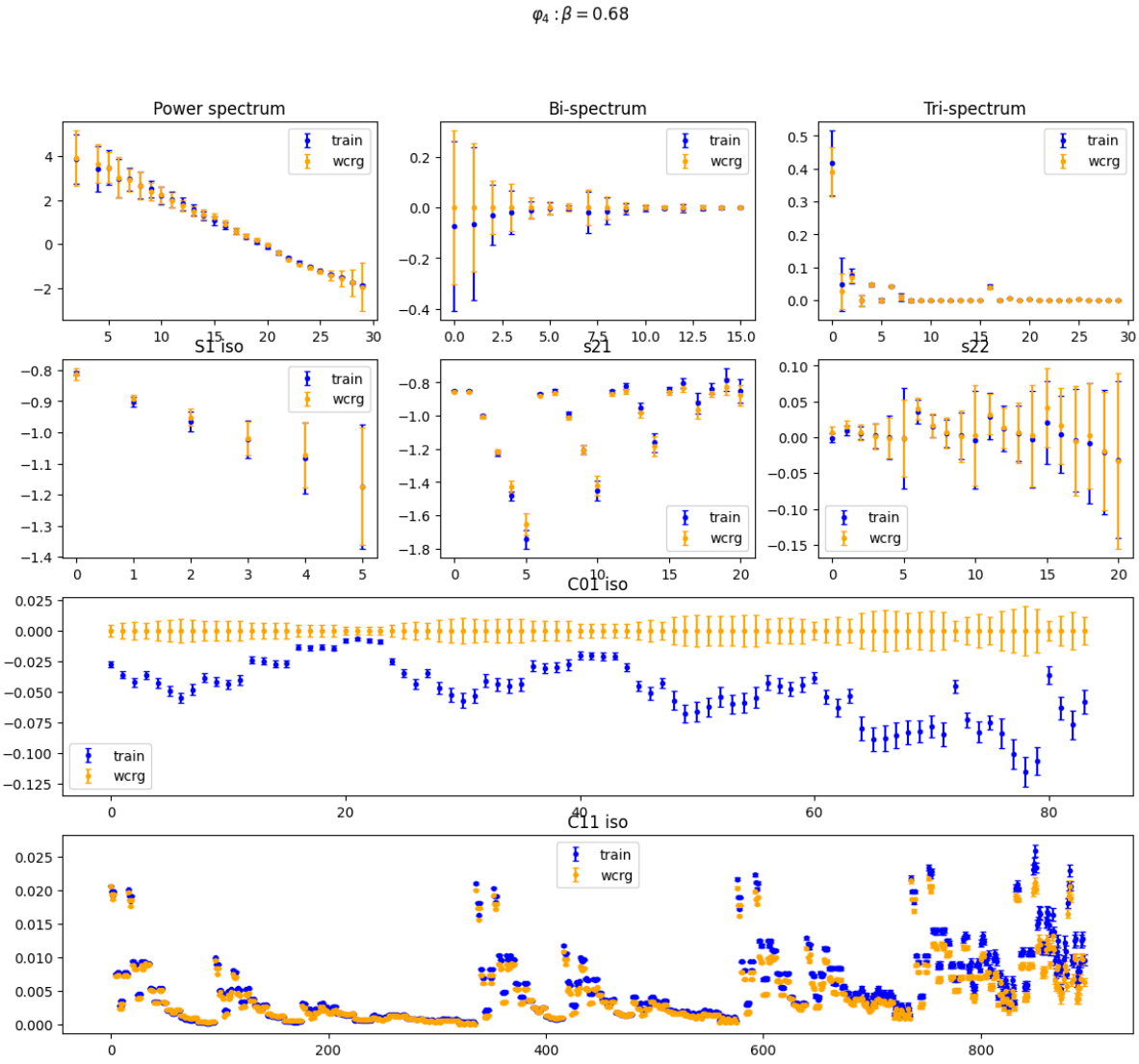

FIGURE 17 – Statistiques de la figure 14 mais concernant les données ϕ_4 ($\beta = 0.68$). Notons que l'on dispose de 500 cartes "train" et de 500 cartes "wcrg".

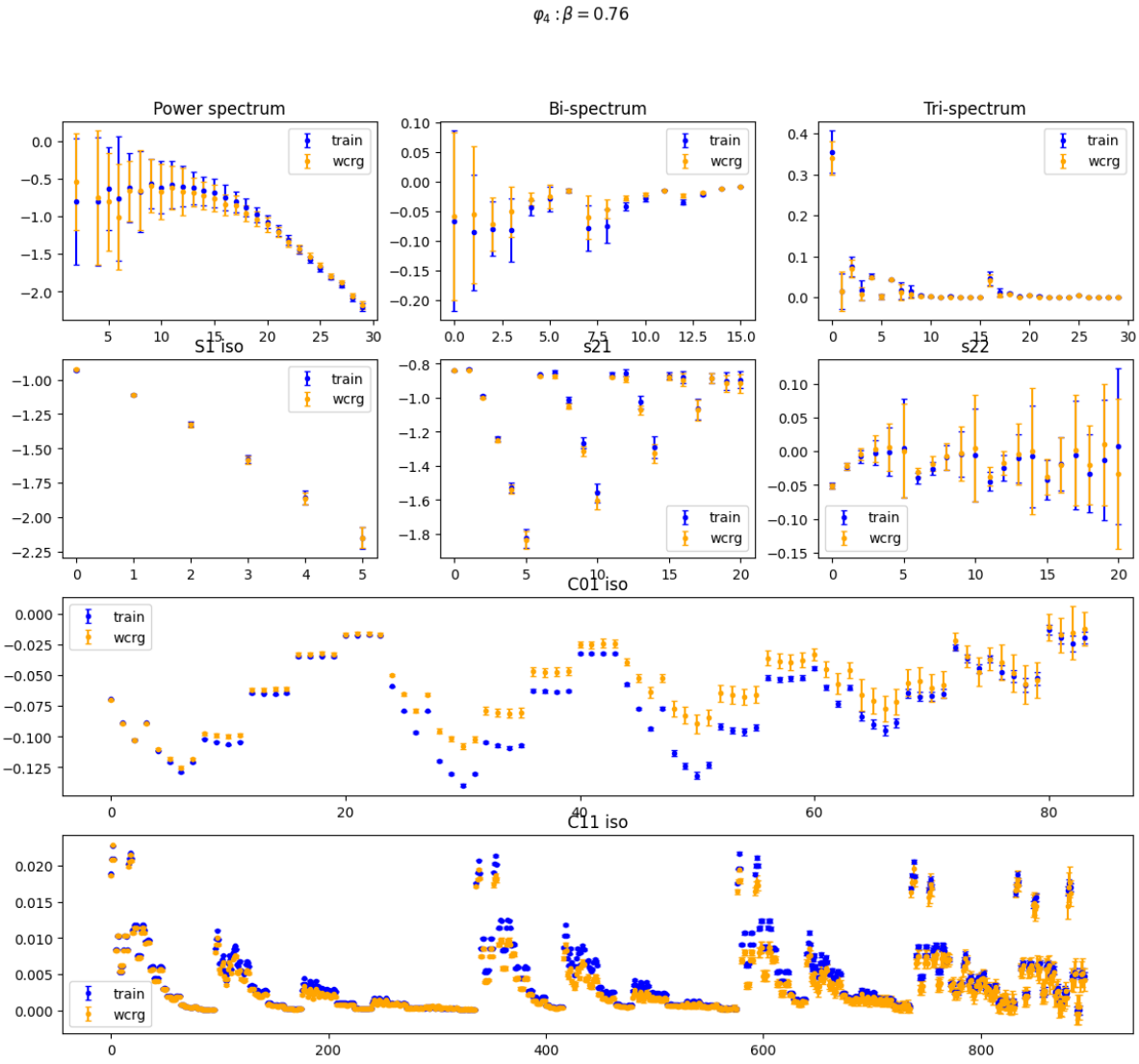

FIGURE 18 – Statistiques de la figure 14 mais concernant les données ϕ_4 ($\beta = 0.76$). Notons que l'on dispose de 500 cartes "train" et de 50 cartes "wcrg".

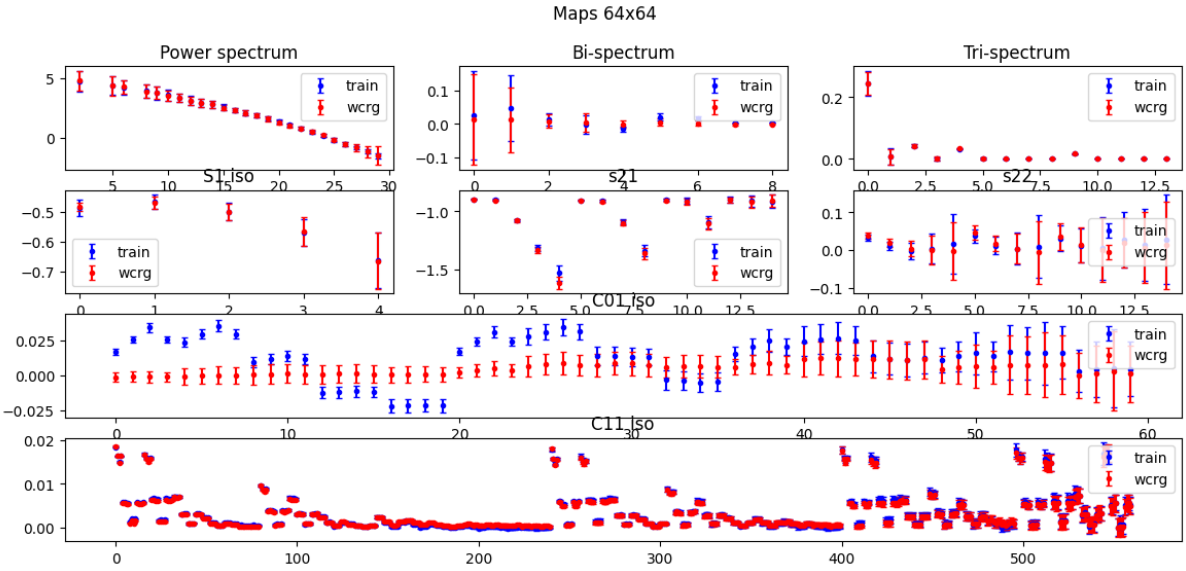

FIGURE 19 – Statistiques de la figure 14 mais concernant les données WL-1 à l'échelle $L = 64$. Notons que l'on dispose de 500 cartes "train" et de 50 cartes "wcrg".

on n'exclut pas un bug sous-jacent, les distributions de ces coefficients générés montrent tout de même des structures sauf précisément pour ϕ_4, $\beta = 0.50$.

Deux expérimentations ont été menées pour essayer de comprendre la distribution des C_{01}^{iso}:

- la première concerne les données WL-1 mais avec les cartes à l'échelle $L = 64$. Le résultat est présenté sur la figure 19. Les désaccords des distributions C_{01}^{iso} sont présents. Notons que les autres distributions sont très bien reproduites.
- la seconde reprend la synthèse avec un modèle micro-canonique mais où l'on a volontairement fixé à 0 les coefficients C_{01}. Le résultat est illustré sur la figure Au-delà de la distribution des C_{01}^{iso}, l'hypothèse d'une corrélation des désaccords C_{01}^{iso}, bi-spectrum et s_{21} peut sembler avoir un certain sens.

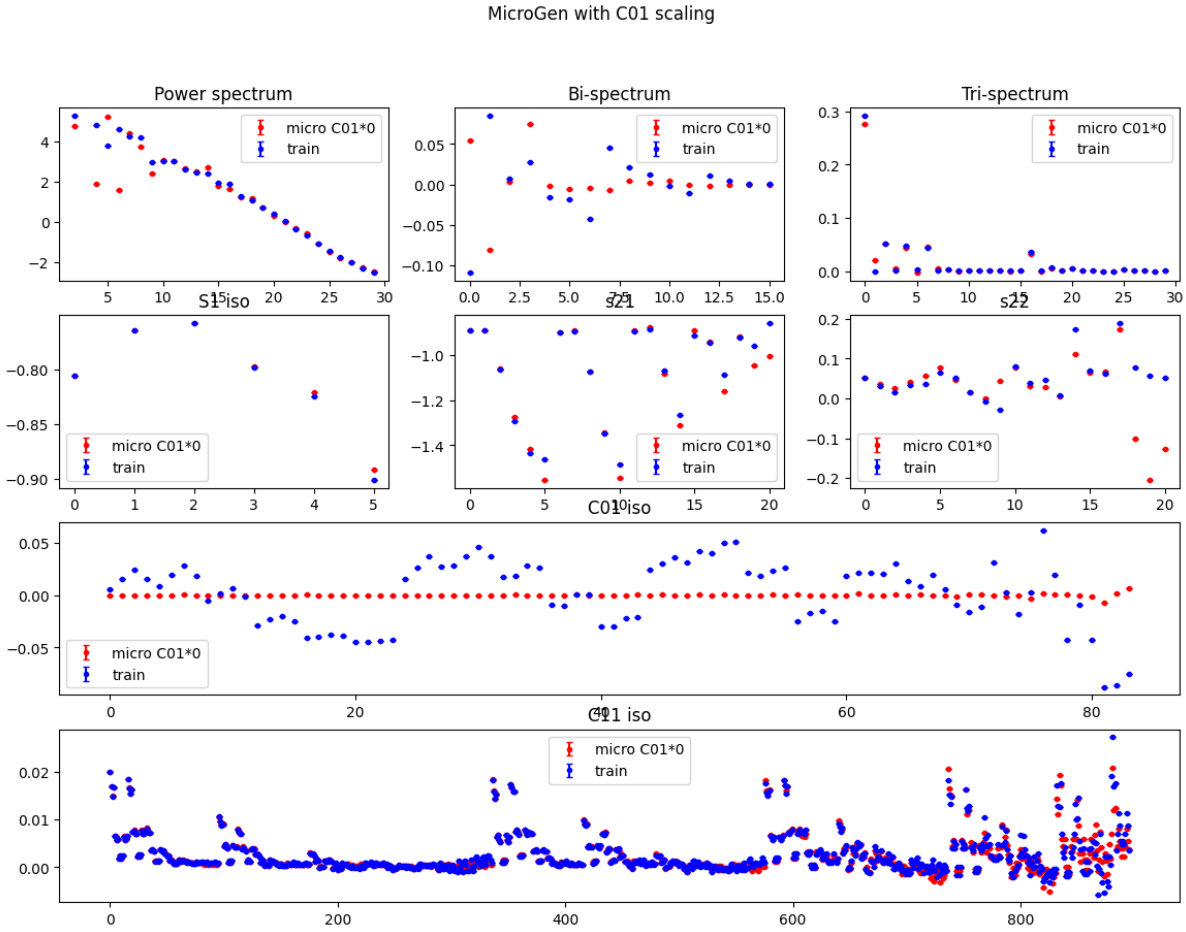

FIGURE 20 – Statistiques de la figure 14 mais concernant les données WL-1 et un modèle micro-canonique du type de celui discuté à la section 4 mais où les coefficients C_{01} sont volontairement mis à 0. Notons que l'on a utilisé qu'une unique carte pour calculer les distributions.

6. Conclusion

Dans cette note, nous avons exposé les résultats de calculs de statistiques permettant d'aller au-delà de l'aspect visuel et spectre de puissance, afin d'apprécier la qualité des cartes générées par des modèles WCRG dans le cas de données de Weak Lensing et ϕ_4. Ce qui en ressort à l'heure de l'écriture de cette note, c'est que si globalement les distributions d'ordre supérieur sont bien reproduites, ce qui est un résultat nouveau très encourageant, il en reste une qui fait défaut à savoir celle des coefficients de corrélations [12] C_{01}. Sauf en fait dans un cas particulier, à savoir ϕ_4 avec $\beta = 0.50$, mais il se trouve que la distribution des C_{01} obtenue sur les cartes d'entrainement est nulle.

Donc, il "reste" à comprendre comment on peut améliorer la modélisation des potentiels WCRG afin d'obtenir de meilleurs résultats. Notons qu'un réseau DGAN optimisé sur les données WL-1 donne des cartes dont les statistiques sont en parfait accord avec celles obtenues sur les données d'entrainement. Il en va de même avec des modèles micro-canoniques mais qui par nature sont optimisés de telles distributions.

12. Notons que nous n'avons pas montré les distributions les distributions des coefficients "non-iso" mais nous avons vérifié que les désaccords sont identiques à ceux observés dans cette note.

Références

Cheng, S. & Ménard, B. 2021, arXiv e-prints, arXiv:2112.01288

Cheng, S., Morel, R., Allys, E., Ménard, B., & Mallat, S. 2023, arXiv e-prints, arXiv:2306.17210

Guth, F., Lempereur, E., Bruna, J., & Mallat, S. 2023, arXiv e-prints, arXiv:2306.00181

Lanzieri, D., Lanusse, F., Modi, C., et al. 2023, arXiv e-prints, arXiv:2305.07531

Mustafa, M., Bard, D., Bhimji, W., et al. 2019, Computational Astrophysics and Cosmology, 6, 1