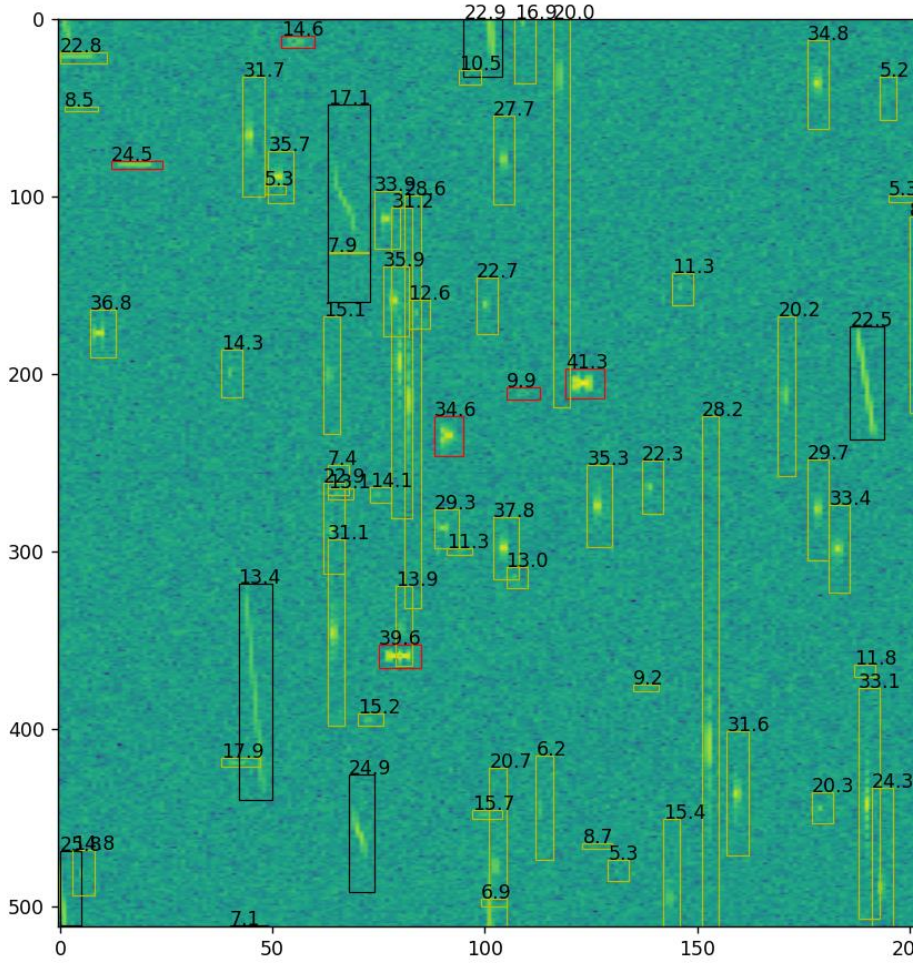
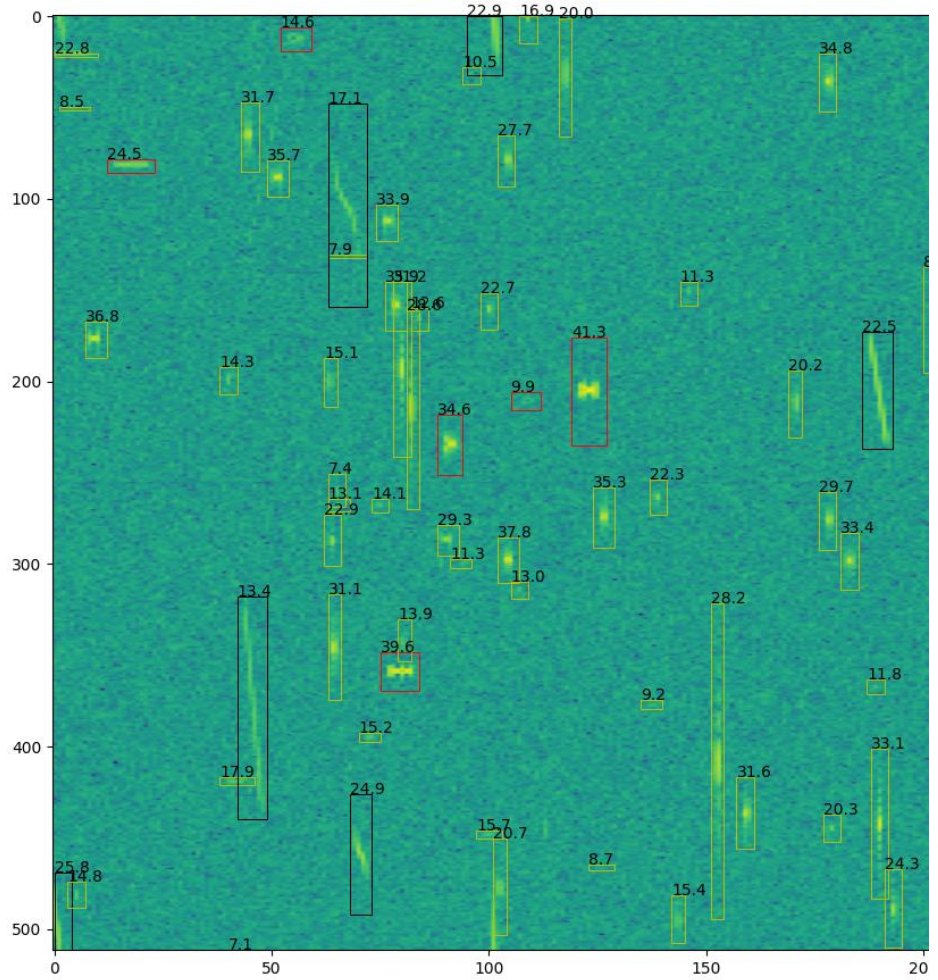
Extraction de caractéristiques

Nouveau fenètrages en fréquence

avant



après



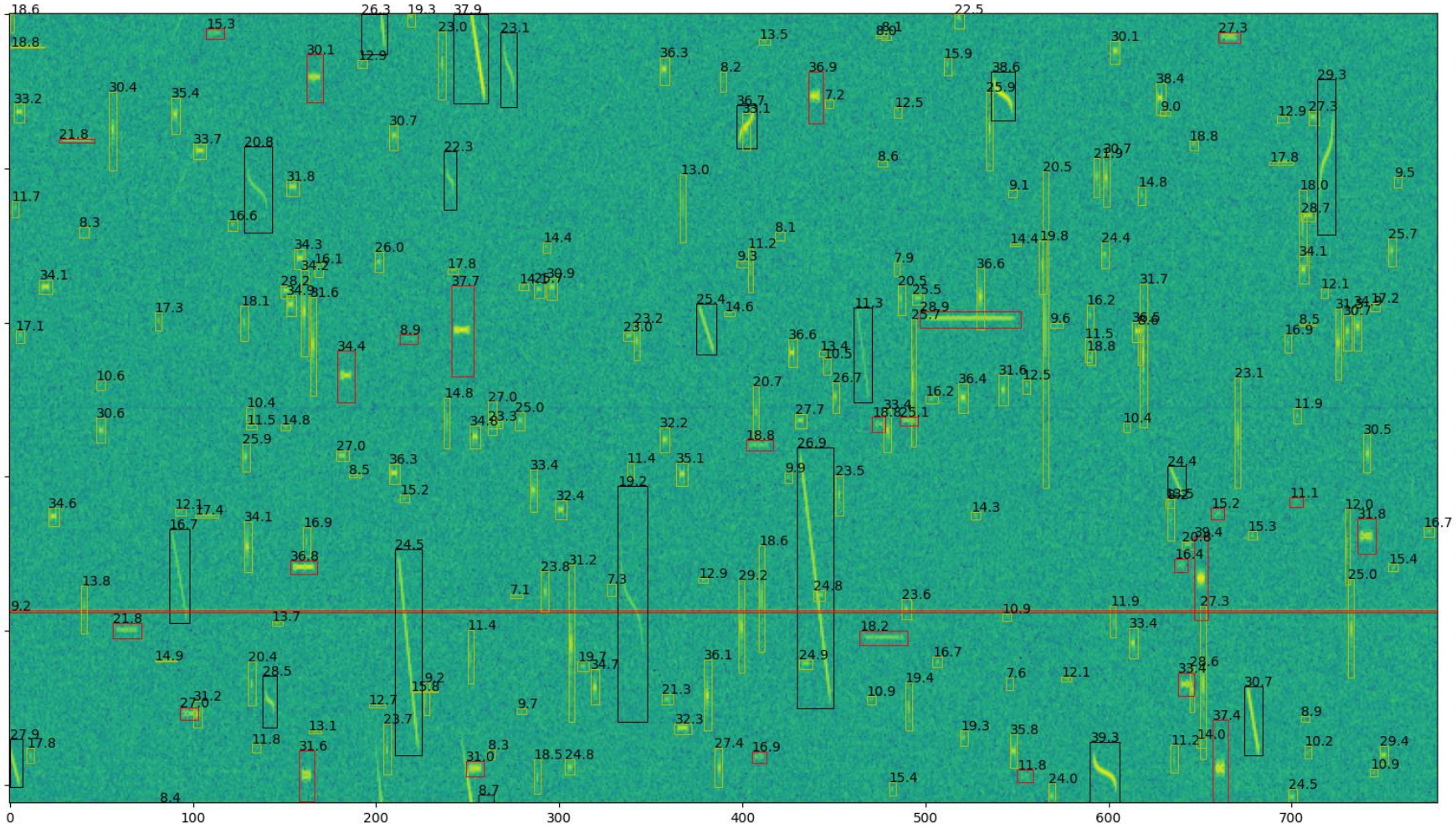
Plus de sinc->on regarde le gain de la fft de la fenêtre de tukey qu'on utilise

Pour les p mods et no modson regarde à partir de quel décalage de fréquence la DSP arrive au niveau du bruit

modèle de DSP pour les no-mods, DSP de réf pour les p-mods

Pour les chirps, on garde la bande de réf

Signal de test, fs_rec = 1Go, T = 2e-4s

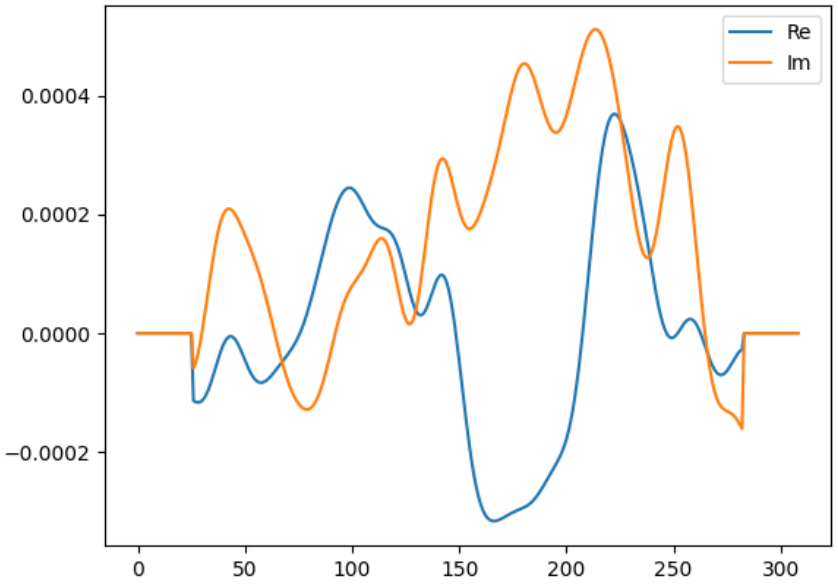


On ne regarde pas les signaux dont les box atteignent les bords

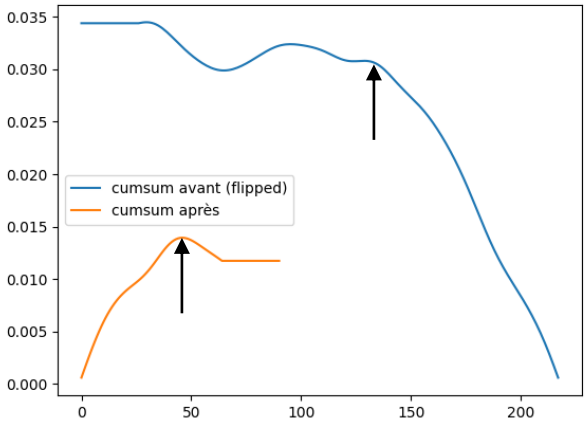
no-mods

- On recalcule la fréquence centrale
- On filtre autour de la bande avec un FIR
- On se place en le argmax du module
- On se déplace à droite et à gauche et on calcule les cumsum
- Quand le cumsum n'augmente plus, fin de PW

Signal filtré $\sigma_{bruit\ filtré} = 2e-4$
7,5dB



On voit le signal 18 samples trop tot
On le voit durer 135 samples à la place de 99



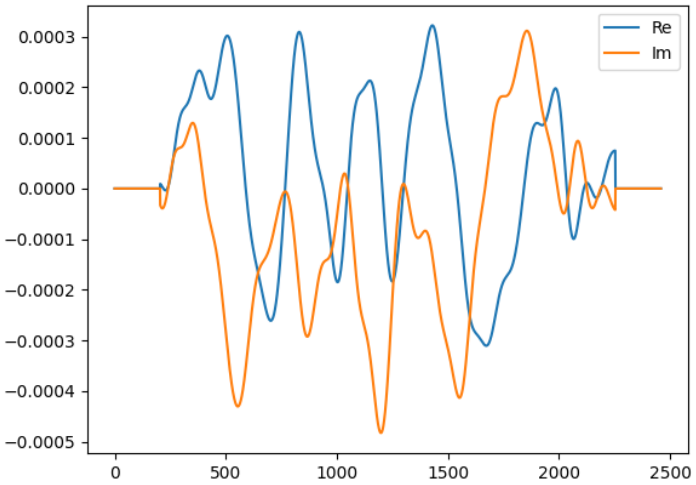
Equivalent à trouver la fenêtre qui maximise la fft en 0
On peut affiner un peu mais c'est correct
Il faudrait peut-être mieux utiliser un IIR, mais pas en simu

Détection des p-mod

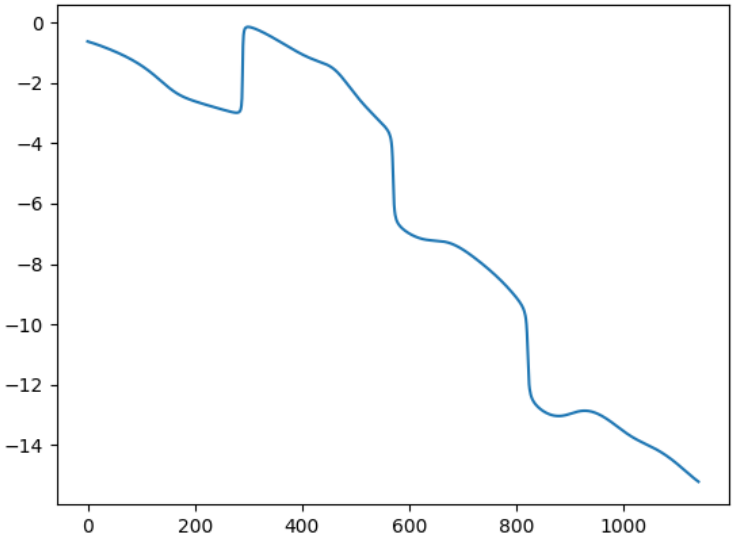
- On fait l'hypothèse que le signal dépasse $3\sigma_{bruit\ filtré}$ sinon phase trop bruitée
- On recalcule la fréquence centrale
- On calcule le unwrap de la phase du signal> $3\sigma_{bruit\ filtré}$
- On regarde les outliers dans sa dérivée
- Seuil pour outliers: 8 fois la médiane

L'approche nous restreint aux forts SNR
On pourrait faire mieux si il n'y avait que des sauts de 180°

Signal filtré $\sigma_{bruit\ filtré} = 1{,}06e-4$
15,3dB



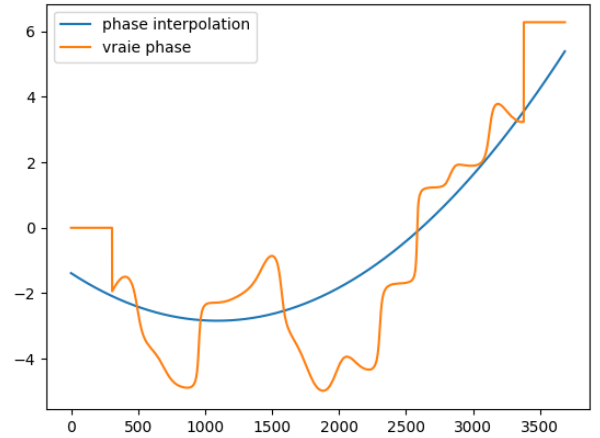
phase



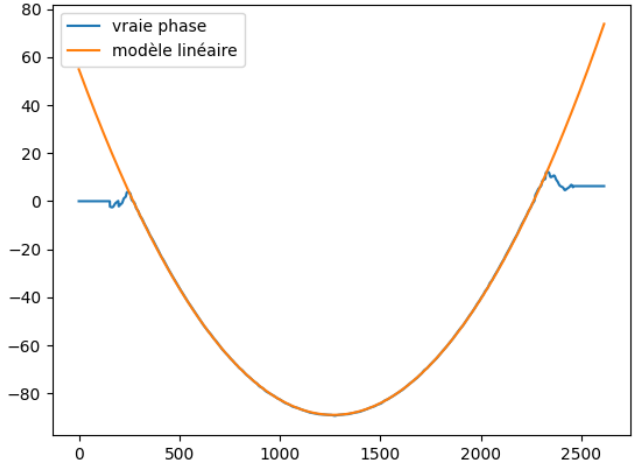
Traitement des fmods

- Pas d'hypothèse sur l'amplitude
- On fit la phase du signal filtré avec les modèles
- Pondération par le module
- Pas fini
  - Pas de fit slaw
  - Pas de détection de bord

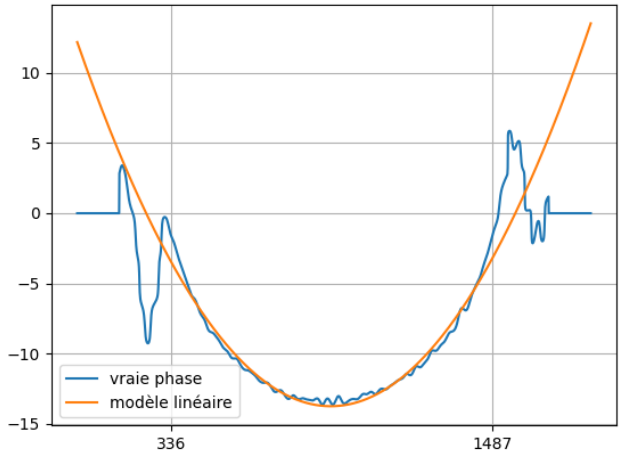
8,4dB (linéaire bord de bande)



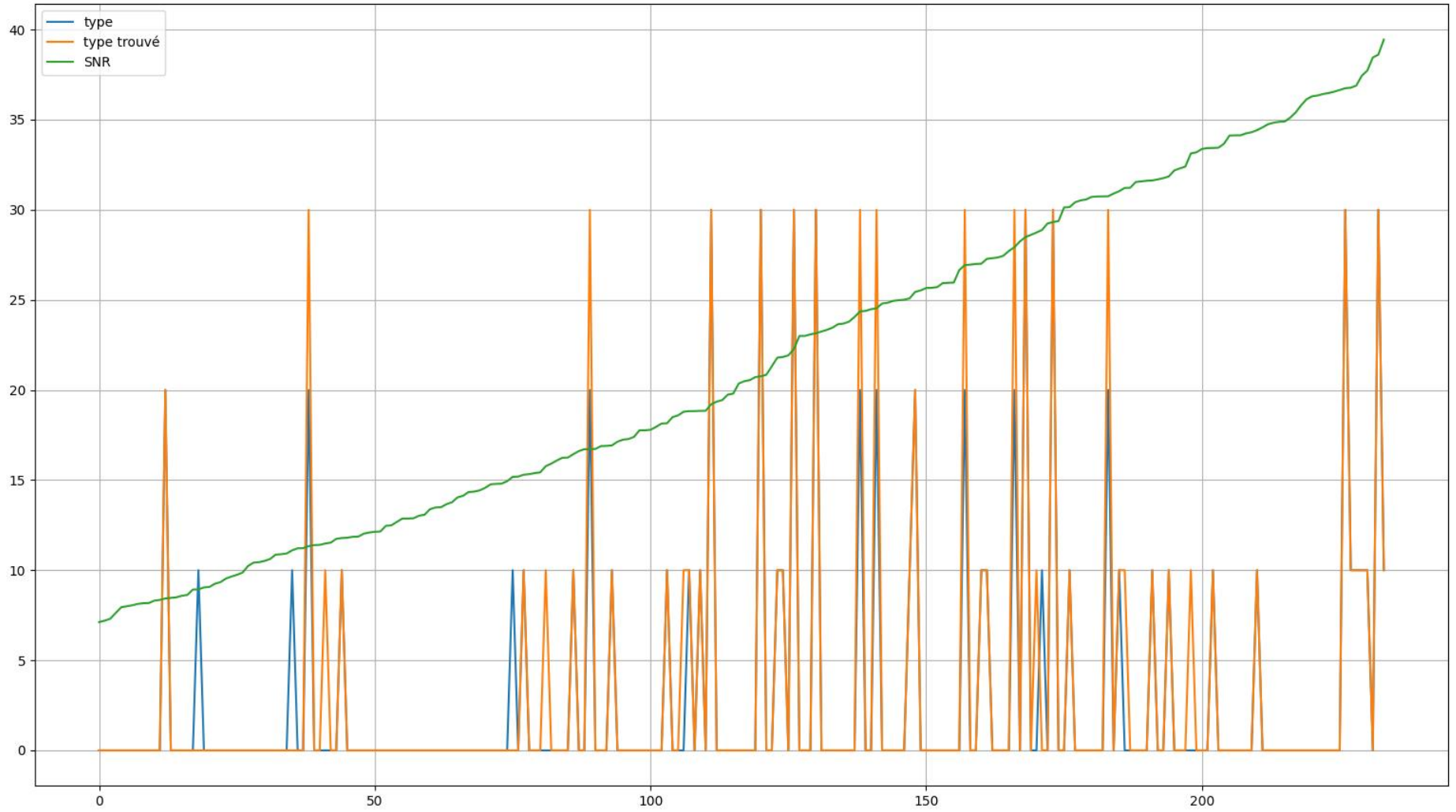
25,4dB



28,5dB slaw

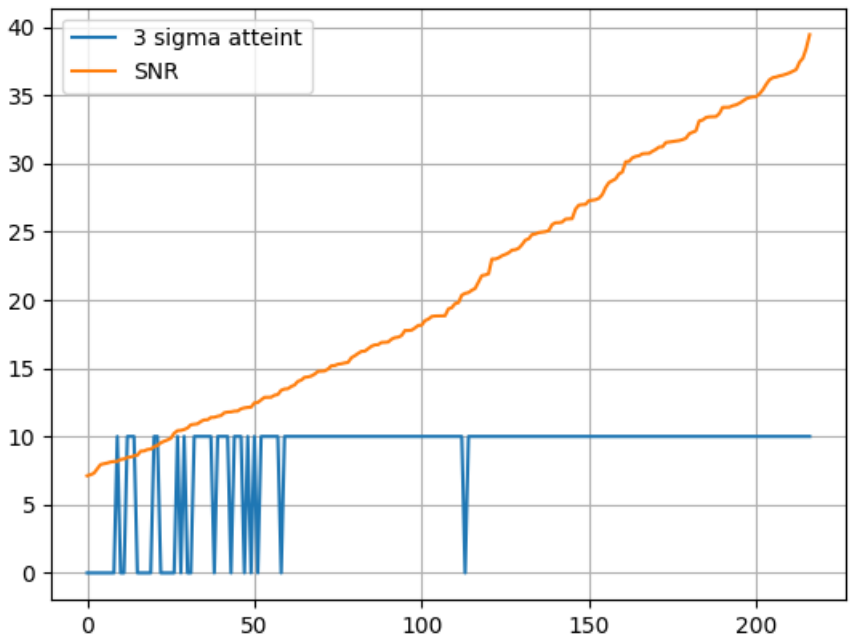


On gagnerait sans doute à faire des fft fractionnaires (lourd si on connait pas la pente)
Il faut voir si le traitement de base suffit pour les SNR qu'on arrive à détecter

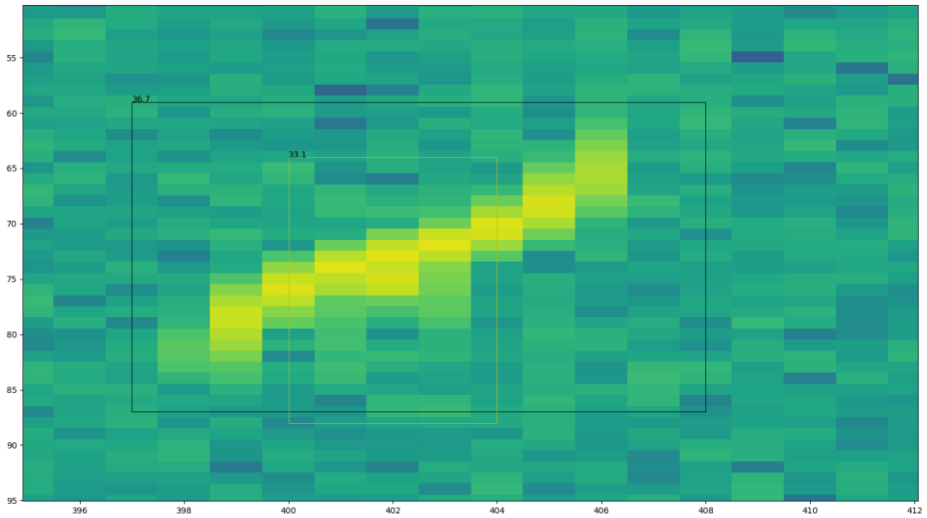


0-> no mod, 10 -> p_mod, 20 30-> linear/slaw f_mod (pas fini)

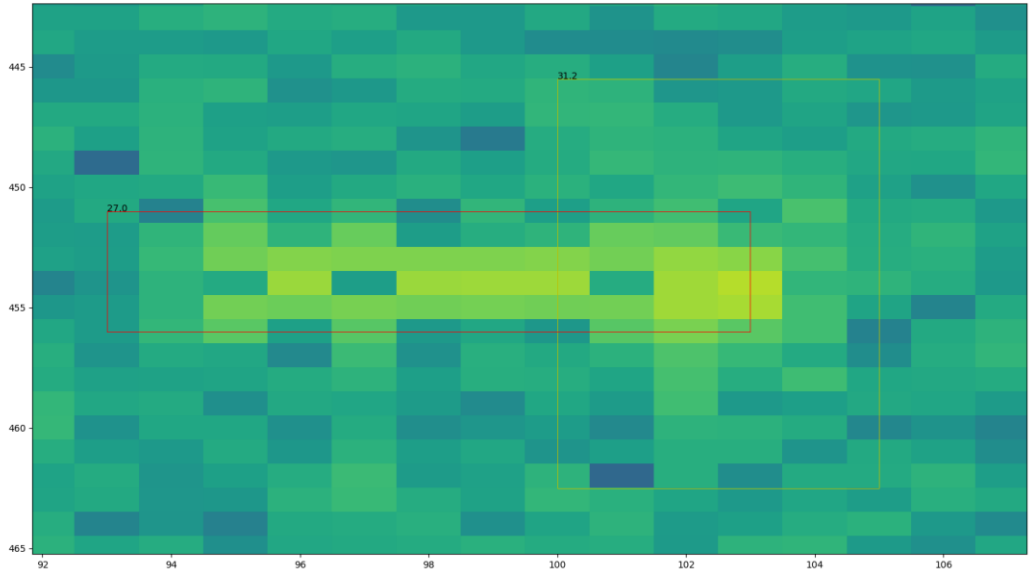
On trouve bien tous les no mod en faible SNR mais on n’a pas de mérite



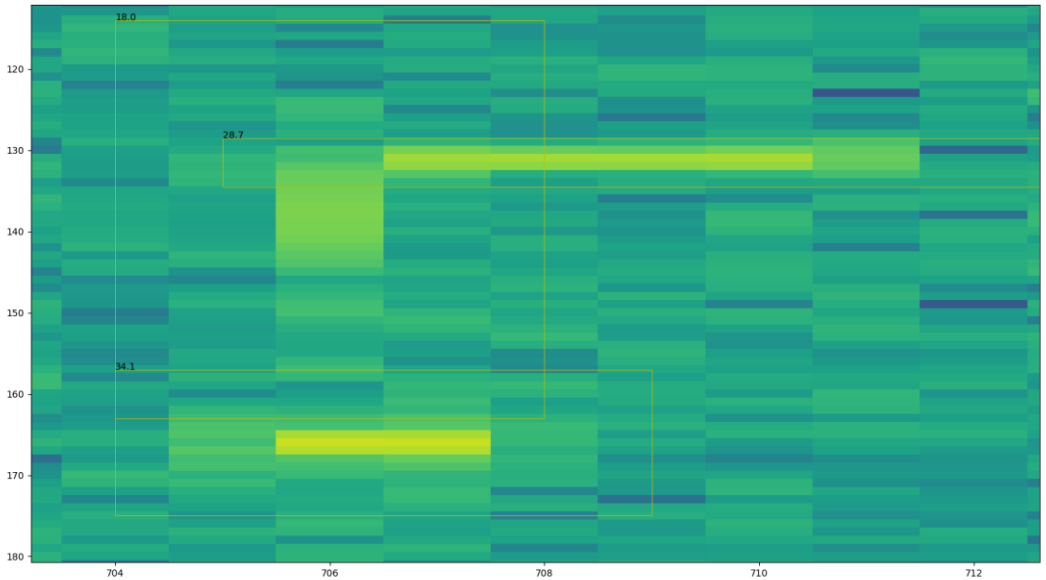
198



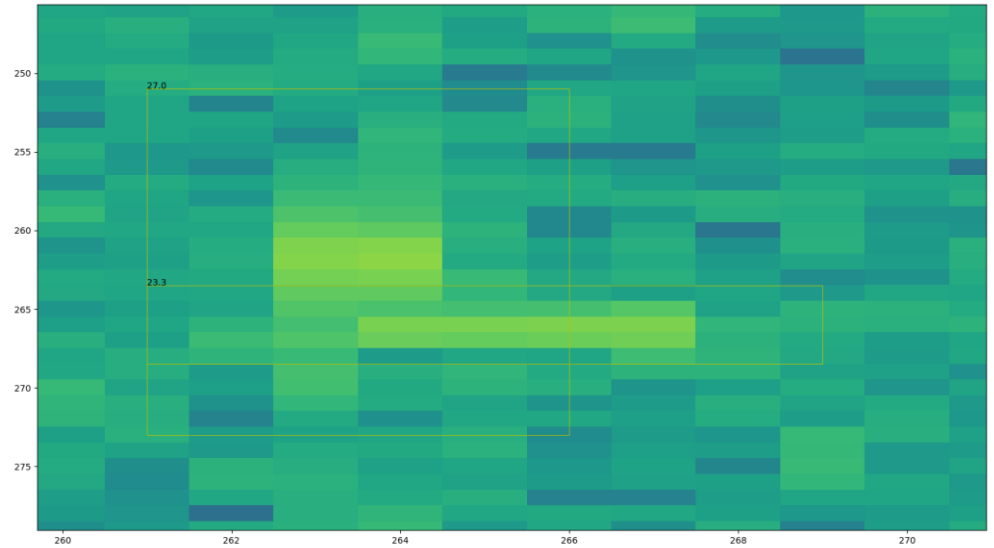
186



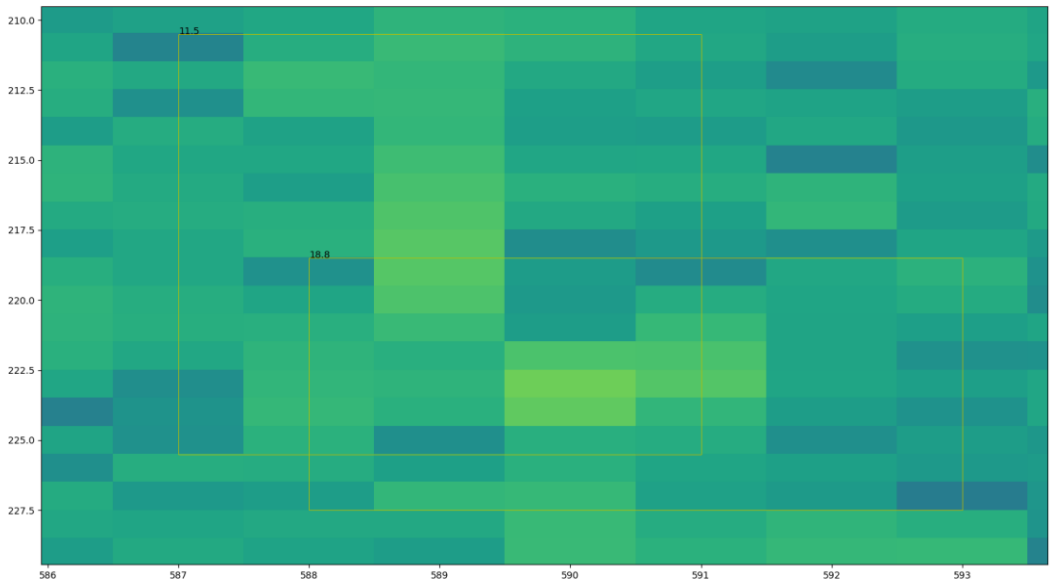
170



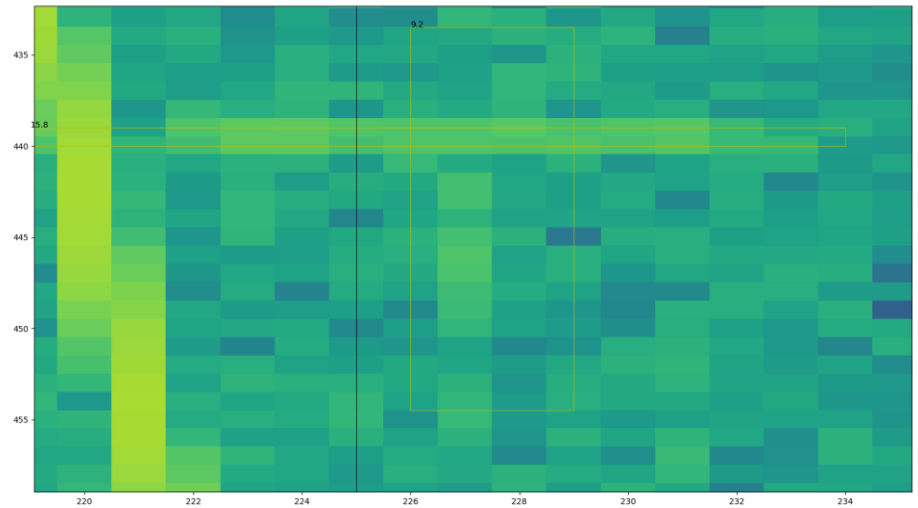
132



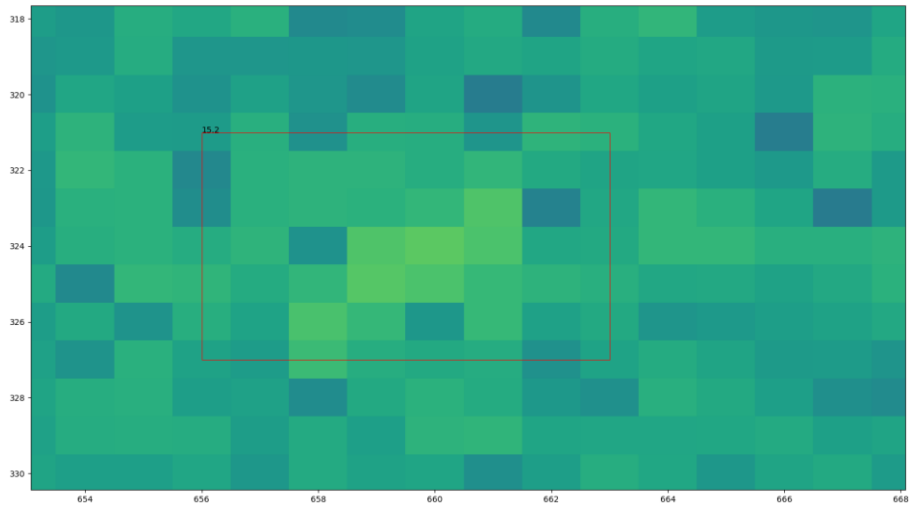
106


11.5
18.8
210.0
212.5
215.0
217.5
220.0
222.5
225.0
227.5
586
587
588
589
590
591
592
593

81

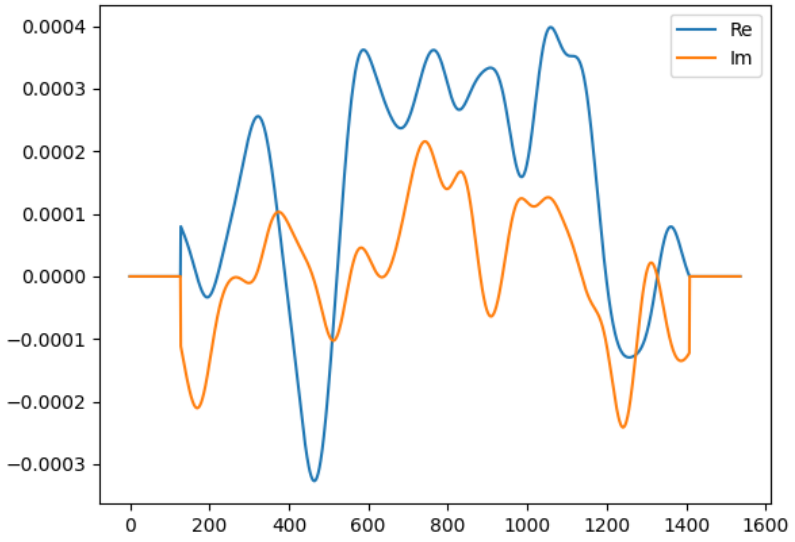

9.2
15.8
435
440
445
450
455
220
222
224
226
228
230
232
234

75 pas d’interférences 15dB




```
labels['mod']['sequence']
> array([1., 0., 1., 1., 1., 1., 0.])
```

Signal filtré



σ du bruit filtré de 1,1^{e}-4

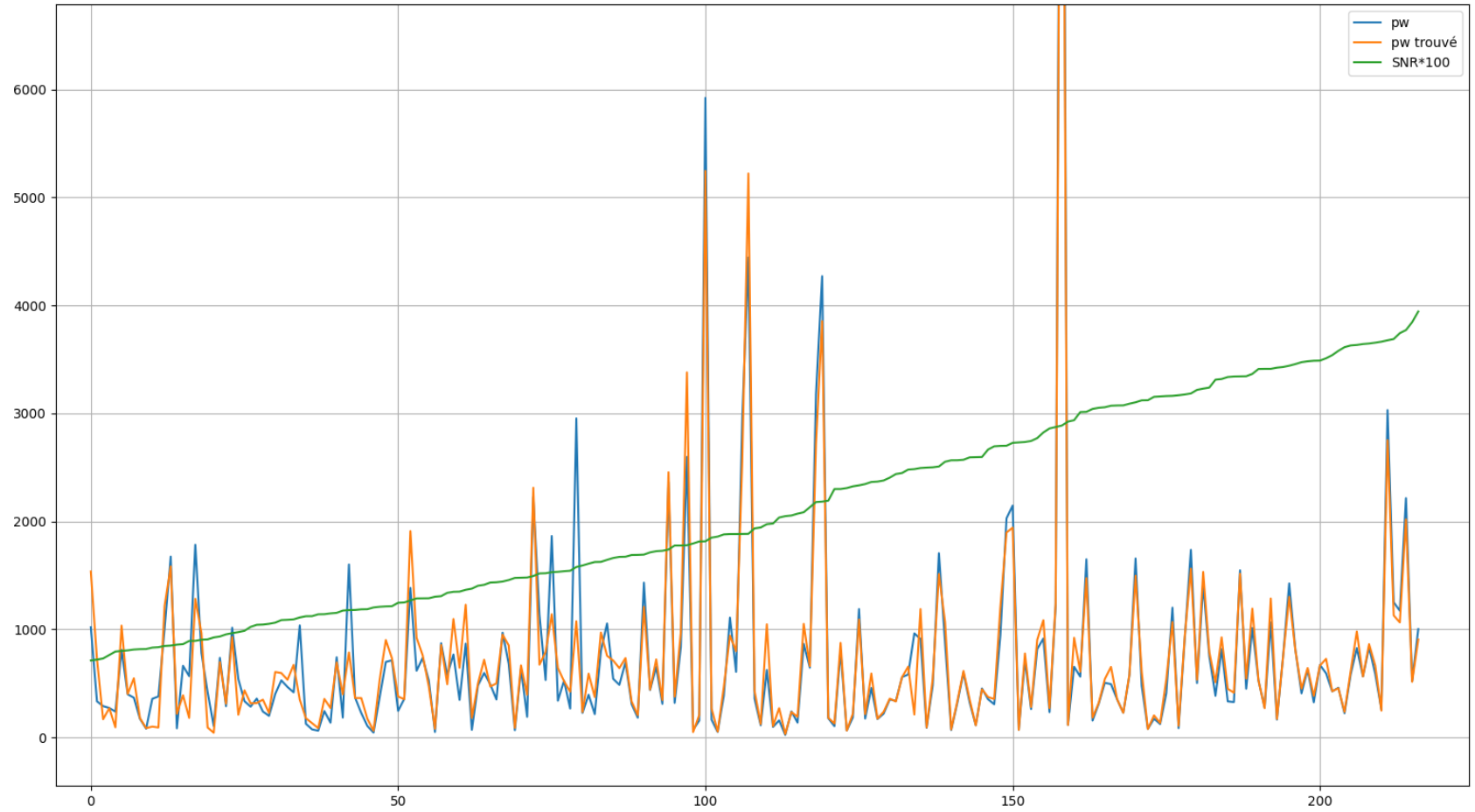
Seule la phase du milieu sort du bruit

Peut être à cause du fenêtrage ou du temps de montée du filtre

Commentaire sur la détection du type

- Si la détection des pmods en bas SNR n'est pas assez bien, on peut repenser l'algo (extraction de la porteuse, corrélations)
- Pas facile, la valeur du SNR est trompeuse
- On ne vérifie pas que l'angle est $\in [45°, 60°, 180°]$ -> permettrait de voir des no mod mal étiquetés
- Cas pmod 45° et 60° compliqué

Pw hors chirps



Quelques outliers mais OK

Il faut se souvenir qu’il y a le temps de montée des filtres et la fenètre de tuckey des signaux

Reste à

- Calculer le nombre de shifts des pmods (et leurs positions)
- Calculer l’angle des pmods
- Finir les fmods
- Rassembler les «constantes cachées» et justifier
- Affiner les filtres
- Clean et commit le code