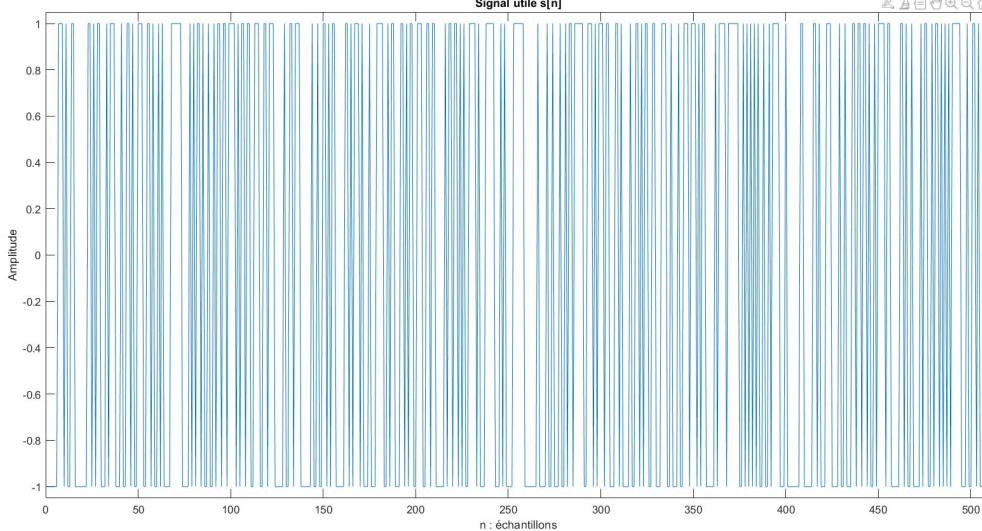


Signal - TP2 - Thème3

Maéva Bachelard - Margot Laleu



Dans un premier temps, nous créons notre signal utile s[n] du type BPSK (constitué uniquement de -1 et de 1) et nous observons son spectrogramme en dB.

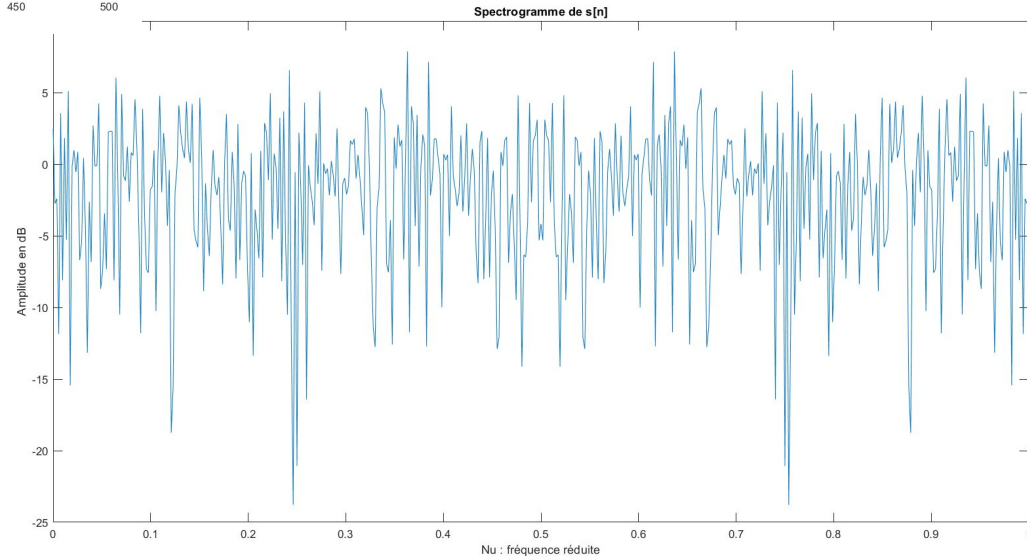
$$I_{XN} = |X_N[k]|^2 / N$$

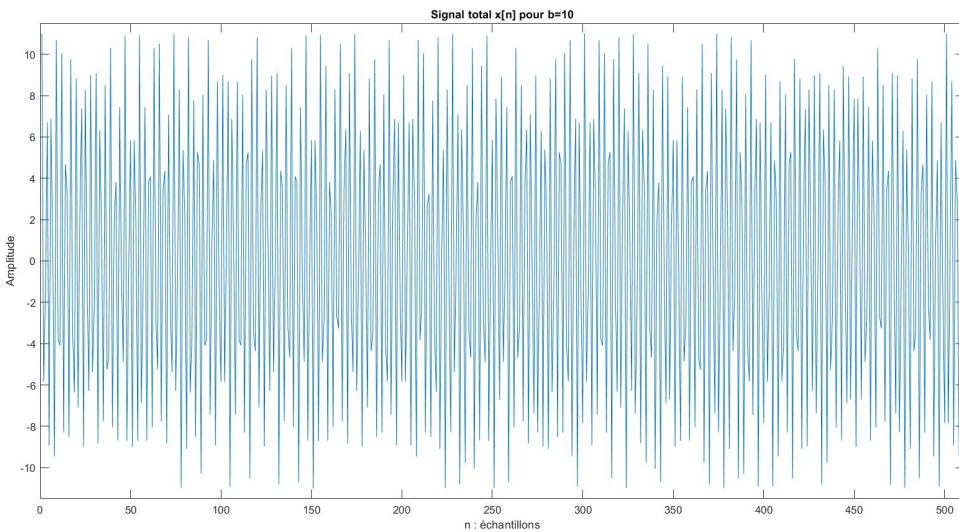
et en dB $I_{XN/dB} = 10 \cdot \log(I_{XN})$

I. Suppression d'un brouilleur bande étroite

1. Génération des signaux

Rappel Matlab : log 10/ln avec “log” sur matlab



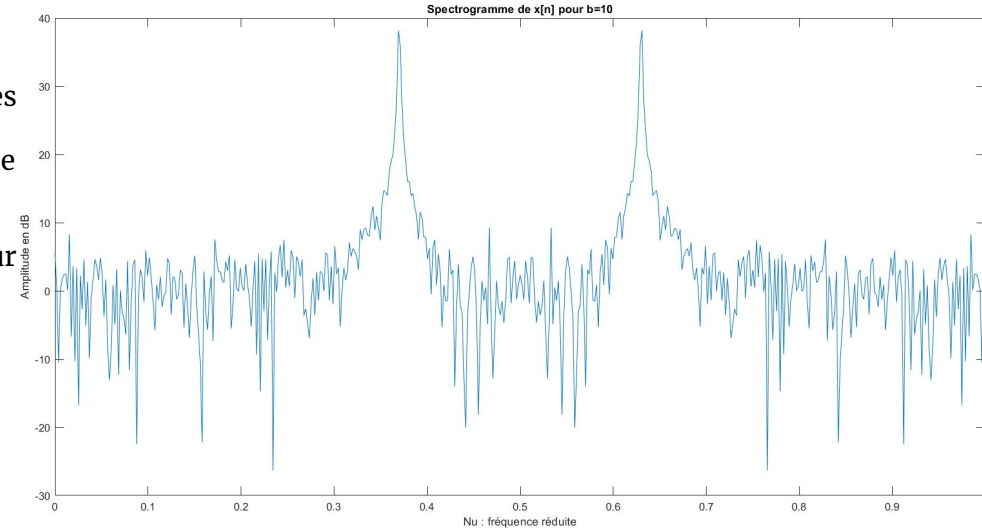


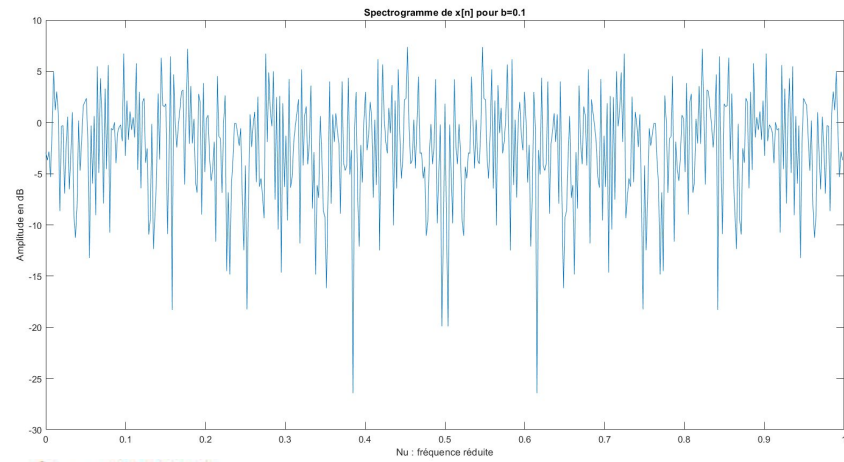
On utilise le bruit créé lors du thème 1 : $b[n] = b \cdot \cos(2\pi\nu_0 n + \varphi)$ avec φ arbitraire et $\nu_0 = 0.37$.

Ensuite, on ajoute ce bruit au signal utile, donnant le signal total $x[n]$ que l'on observe en temporel, ainsi que son spectrogramme.

On remarque que l'ajout du bruit au signal utile ajoute bien 2 raies en fréquentielles, aux fréquences $\nu_0 = 0.37$ et $1-\nu_0 = 0.63$ correspondant bien au spectrogramme de notre cosinus.

On remarque que l'amplitude du bruit étant très importante (pour $b=10$), que le signal utile est noyé dans le bruit.



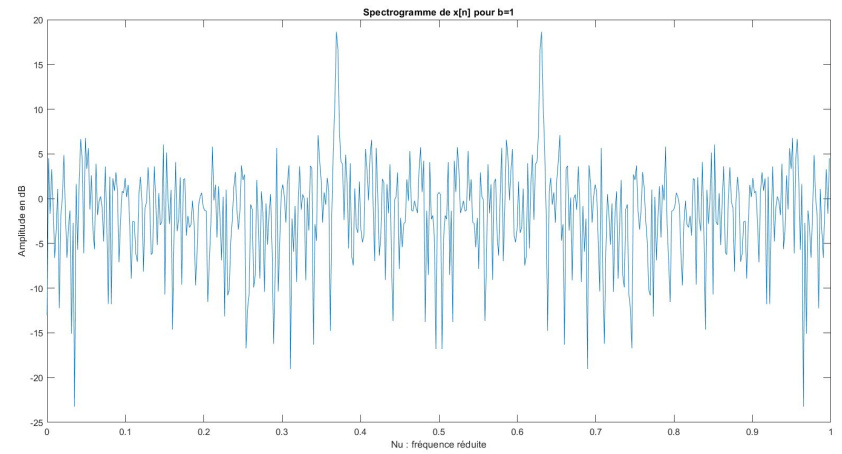


$b = 0.1000$

RSB = 199.8217

RSBdB = 23.0064

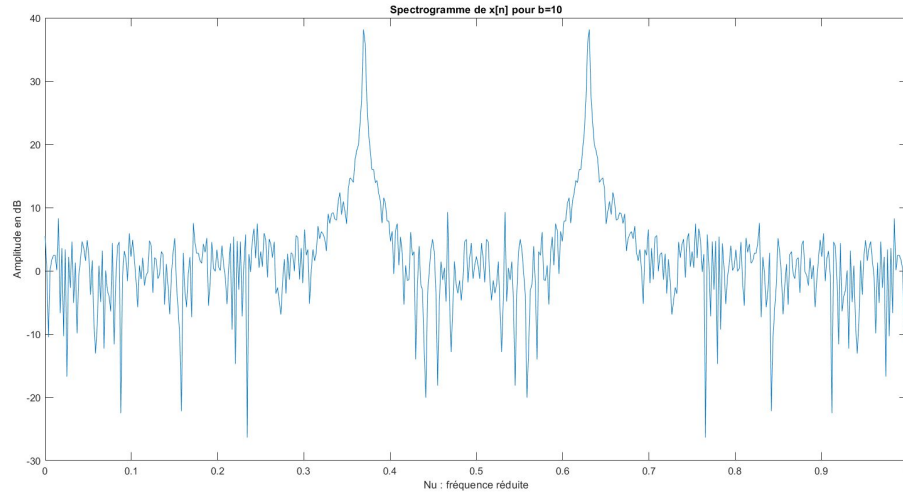
La diminution de l'amplitude du cosinus b , entraîne une diminution de l'amplitude spectrale des raies dues au bruit, on ne les voit même plus, parmi le signal utile, pour $b=0.1$. Ainsi, plus b diminue, plus le RSB est grand, et plus cela est intéressant car le signal est lisible et le bruit négligeable, il faut donc un bruit très faible (un b petit).



$b = 1$

RSB = 1.9982

RSBdB = 3.0064



$b = 10$

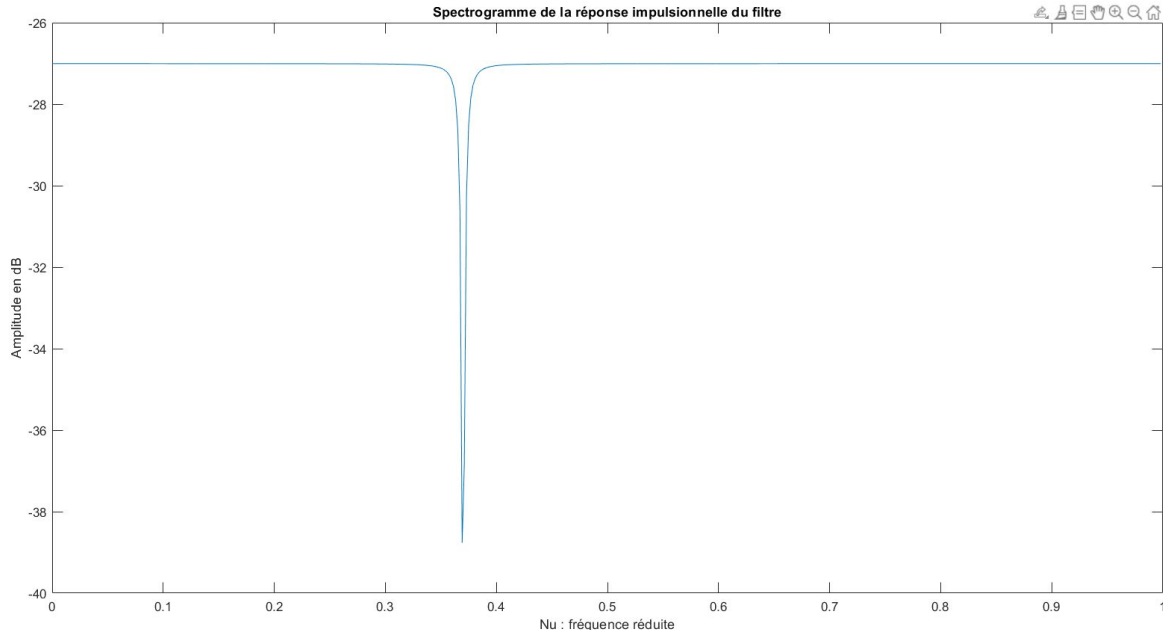
RSB = 0.0200

RSBdB = -16.9936

2. Filtrage coupe fréquence

Le bruit est un cosinus, donc son spectrogramme est constitué de 2 raies en ν_0 et $-\nu_0$ ($1-\nu_0$ sur $[0;1]$). Il faut donc supprimer ces 2 raies et rien de plus, on utilise donc un filtre coupe bande en ν_0 et un second en $1-\nu_0$.

Pour cela, on utilise donc un filtre de fonction de transfert telle que sur la photo ci-dessous.



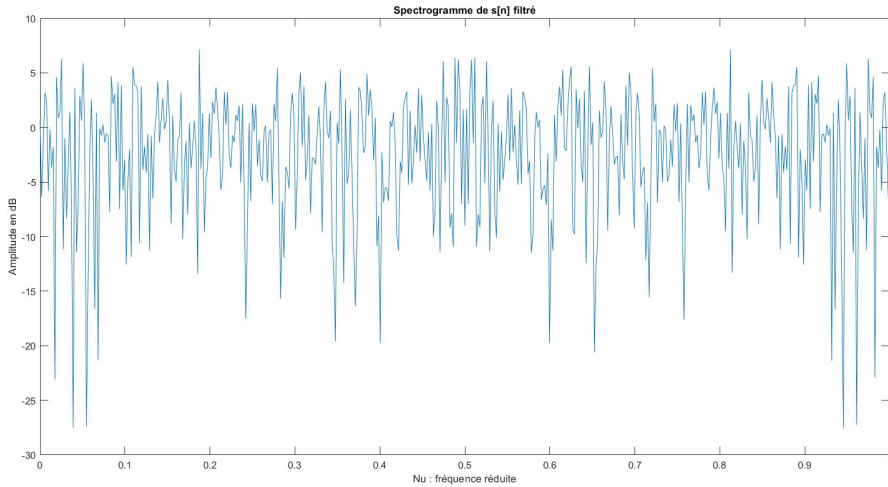
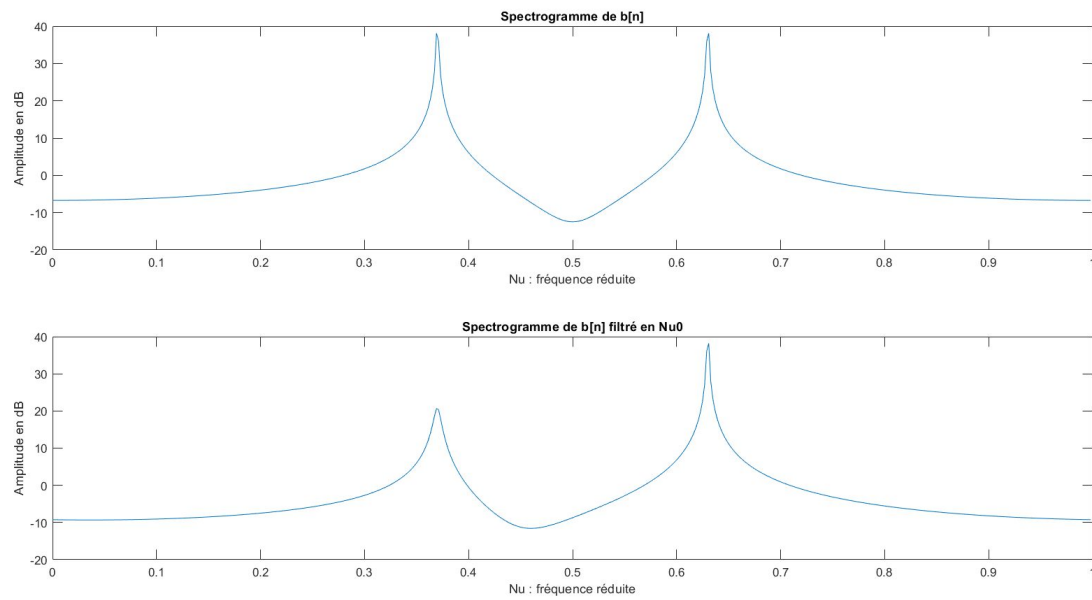
Solution : filtre coupe-bande en ν_0 : $H_{\nu_0}(z) = \frac{1 - e^{2\pi i \nu_0} z^{-1}}{1 - 0.98 e^{2\pi i \nu_0} z^{-1}}$
 $H_{\nu_0}(\nu_0) = 0$ et $H_{\nu_0}(\nu) \approx 1$ ailleurs
Matlab : $y_N = \text{filter}([1, -e^{2\pi i \nu_0}], [1, -0.98 e^{2\pi i \nu_0}], x_N)$

On obtient la réponse en fréquence du filtre, grâce au spectrogramme de la réponse impulsionnelle du filtre, soit la réponse du filtre appliqué à un dirac en entrée.

On applique maintenant notre filtre sur notre bruit seul. Notre filtre n'affectant que la raie en ν_0 , on peut comparer le bruit avec et sans filtrage.

On remarque que le filtre diminue de moitié l'amplitude en dB du bruit, donc corrige partiellement le problème du bruit.

En appliquant notre filtre sur le signal utile, on remarque que le filtre n'altère pas le signal utile.



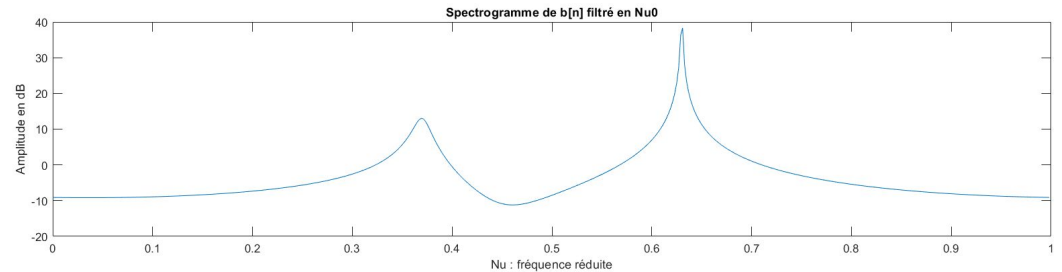
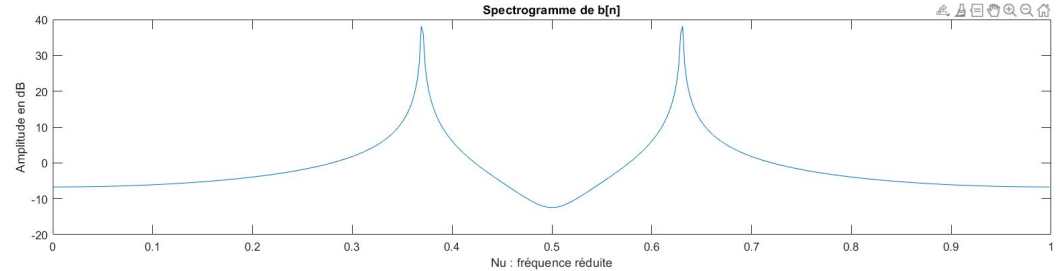
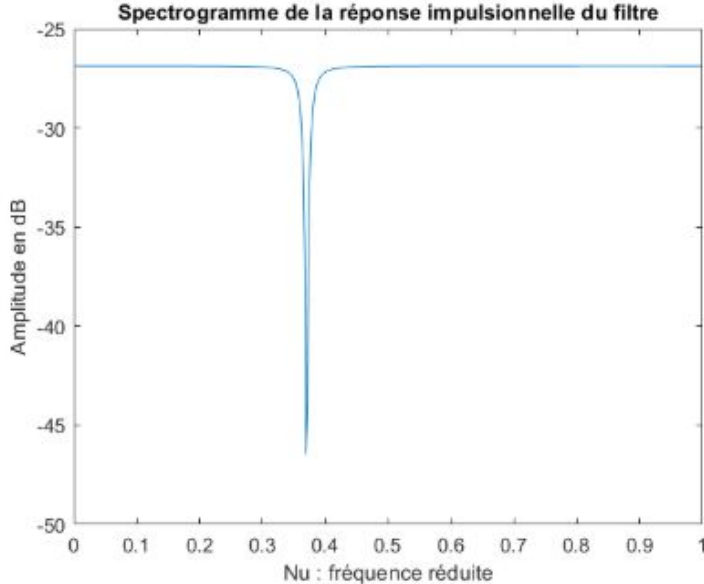
Aux vues de ces résultats, nous allons essayer de diminuer l'amplitude de la raie du bruit après filtrage. Pour cela, dans la formule ci-dessous de notre filtre, nous modifions la valeur surlignée.

```
yb = filter([1, -exp(i*2*pi*Nu0)], [1, -0.98*exp(i*2*pi*Nu0)], bn)
```

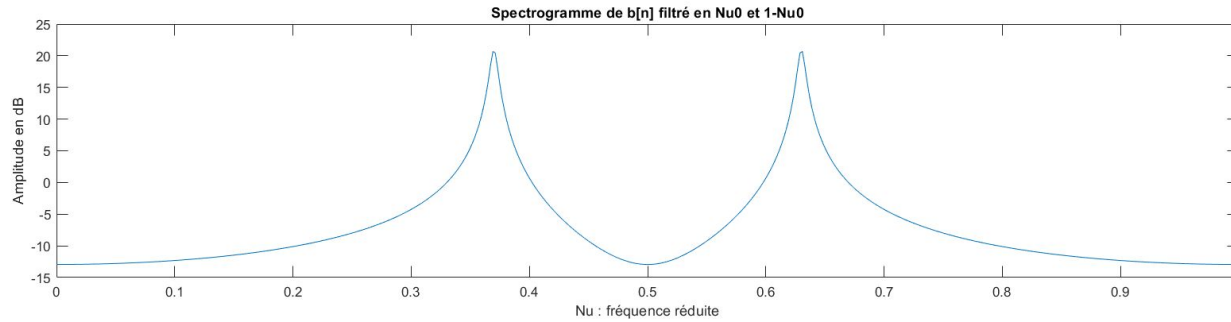
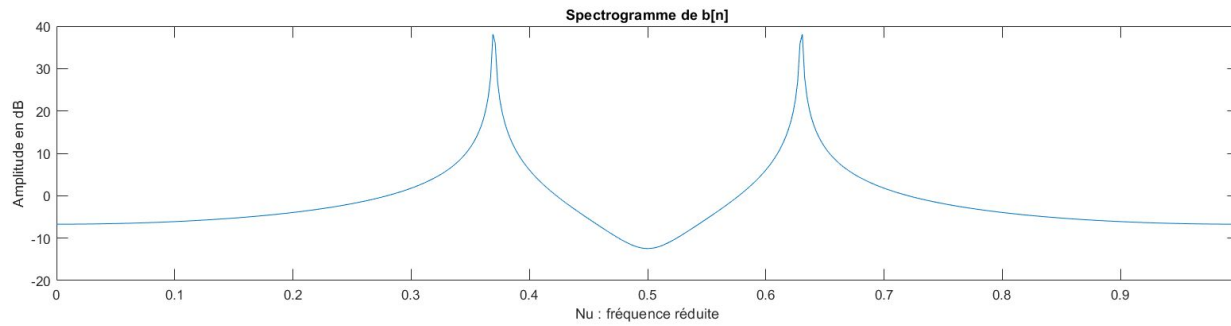
Si nous augmentons la valeur surlignée, cela réduit encore le pouvoir du filtre, donc il faut la diminuer.

Ainsi nous améliorons la capacité de filtrage du bruit du filtre en choisissant 0.95, mais attention, si on diminue trop cette valeur on risque de perdre aussi la partie du signal utile en $Nu0$ et autour de $Nu0$.

Le fait que les valeurs autour de $Nu0$ soient touchées par le filtre vient du fait qu'on a en pas un dirac en réalité mais un sinus cardinal dû au fait que les signaux numérique sont finis, donc agissent comme si on leur appliqué une porte, qui en fréquentielle implique l'apparition d'un sinus cardinal.



```
yb = filter([1, -exp(i*2*pi*Nu0)], [1, -0.95*exp(i*2*pi*Nu0)], bn)
```



Nous créons un second filtre, coupe bande en $1-Nu_0$, afin de filtrer l'autre raie du bruit. Puis nous passons le signal dans nos deux filtres successivement afin d'obtenir un signal filtré des deux côtés.

Par la suite nous allons garder 0,98 et non 0,95 pour ne pas perdre de signal utile.

Ainsi l'amplitude du bruit est passée de 40 dB à 20 dB environ, le résultat n'est donc pas parfait.

$b = 10$	$b = 1$	$b = 0.1000$
$P_s = 1$	$P_s = 1$	$P_s = 1$
$P_b = 50.0446$	$P_b = 0.5004$	$P_b = 0.0050$
$RSB = 0.0200$	$RSB = 1.9982$	$RSB = 199.8217$
$RSB_{dB} = -16.9936$	$RSB_{dB} = 3.0064$	$RSB_{dB} = 23.0064$

Filtrage




$b = 10$	$b = 1$	$b = 0.1000$
$P_{sf} = 1.0249$	$P_{sf} = 1.0013$	$P_{sf} = 1.0237$
$P_{bf} = 2.5159$	$P_{bf} = 0.0252$	$P_{bf} = 2.5159e-04$
$RSB_{dB} = -3.9001$	$RSB_{dB} = 15.9989$	$RSB_{dB} = 36.0948$

La puissance du bruit est divisée par environ 20 par filtrage, le signal est toujours affecté mais cela est moindre (on gagne environ 13dB sur les RSB_{dB} donc cela implique que le signal total est de meilleure qualité).

II. Débruitage d'une image

1. Génération des signaux

Nos images artificielles font 256*256

	ima1	256x256 double
	ima2	256x256 double
	ima3	256x256 double

On crée ensuite un bruit aléatoire 2D multiplié par une constante au carré (nommée sigmab2). On va l'observer pour différentes valeurs de sigmab2 et pour une dimension de la matrice N = 128. On utilise la formule 2D suivante pour calculer les puissances du bruit et du signal utile (ima1 ou ima3) et le RSB_{dB} dans ces différents cas.

Puissance 2D :

$$P_{2D,x} = \frac{1}{M \times N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x[m,n]|^2$$

représentation temporelle du bruit en 2D

spectrogramme du bruit en 2D

sigmab2 = 0.1000

Pb = 0.0098

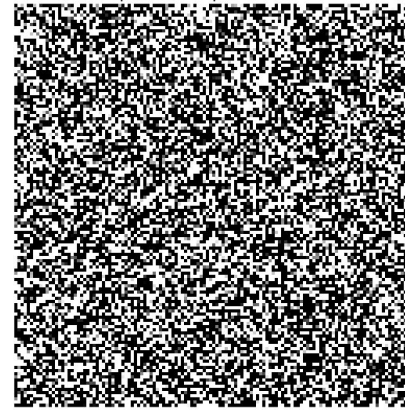
Pim1 = 3.2545e+04

Pim3 = 1.0618e+09

RSBdB1 = 65.2137

RSBdB3 = 110.3494

représentation temporelle du bruit en 2D



spectrogramme du bruit en 2D

sigmab2 = 4

Pb = 16.2426

Pim1 = 3.2545e+04

Pim3 = 1.0618e+09

RSBdB1 = 33.0183

RSBdB3 = 78.1539

Plus sigmab2 est grand (donc plus l'amplitude du bruit est grande), plus le RSB est faible et donc l'image/le signal est difficilement exploitable. On le remarque rapidement sur les images, plus sigmab2 est grand, plus les intervalles entre les nuances de gris sont fortes et donc le bruit impacte beaucoup l'image, inversement un sigmab2 très faible crée un bruit qui impacte moins l'image.

représentation temporelle du bruit en 2D



spectrogramme du bruit en 2D

sigmab2 = 10.0000

Pb = 98.9074

Pim1 = 3.2545e+04

Pim3 = 1.0618e+09

RSBdB1 = 25.1726

RSBdB3 = 70.3082

2. Débruitage par lissage

Un signal $x[n, m]$ 2D, filtré par un filtre $h_{k,l}$ a pour formule :

$$y[n, m] = \sum_{k=-M}^M \sum_{l=-M}^M h[k, l] x[n-k, m-l]$$

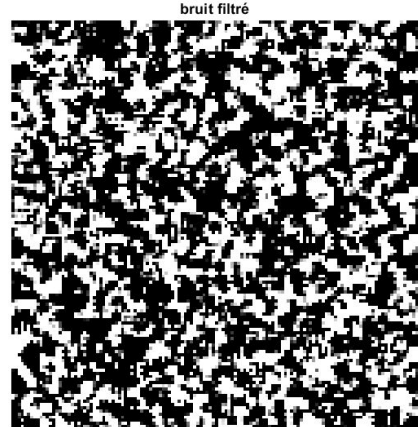
Le masque du filtre $h_{k,l}$ correspond à sa matrice (L, C), ici telle que $L = -M, \dots, 0, \dots, M$ et $C = -M, \dots, 0, \dots, M$.
On souhaite étudier le cas particulier où $M=1$ nous avons alors notre masque qui est de taille 3x3.

Ensuite, nous filtrons nos images et le bruit (séparément)
avec Hmoy un filtre moyennneur de matrice ci-contre.
On obtient alors les résultats suivant :

$$H_{moy} = 3 \times 3$$

0.1111	0.1111	0.1111
0.1111	0.1111	0.1111
0.1111	0.1111	0.1111

Le filtre moyennneur, comme son nom l'indique, a pour effet de remplacer chaque pixel de notre image par la valeur moyenne du pixel et des pixels qui l'entourent. Pour le bruit filtré, on remarque que les zones en noir ont comme étaient "étalées".



Le filtre ne semble pas avoir d'effet sur le haut de ima1, ce qui semble logique puisque ce qui peut être affecté, ce sont uniquement les limites entre les zones noires et blanches. La moyennes de pixels blanc est blanche, idem pour le noir.
Pour le bas, on observe une inversion des colonnes blanches et noires.

