Filtrage numérique avec GNU Radio : approche expérimentale

Culture Sciences de l'Ingénieur

La Revue 3E.I

école normale supérieure paris—saclay

Thomas LAVARENNE[1]

Édité le 08/06/2026

[1] *Enseignant de Sciences Physiques en BTS CIEL option IR – Académie de Créteil.*

Cette ressource fait partie du N°120 de La Revue 3EI du 3ème trimestre 2026.

Le filtrage numérique constitue un élément central du traitement du signal moderne. Il est présent dans un grand nombre de systèmes, allant des communications radio aux applications audio, en passant par l'instrumentation et l'analyse de données. Malgré cette importance, son enseignement reste souvent très théorique, ce qui peut rendre difficile l'appropriation concrète des concepts.

GNU Radio permet de dépasser cette difficulté. Cet environnement offre la possibilité de manipuler des signaux en temps réel, de visualiser leur évolution et de tester directement l'effet des paramètres d'un filtre. Cette approche expérimentale permet de relier de manière immédiate les équations de filtrage à leur impact réel.

Dans cet article, nous proposons une progression basée sur des manipulations simples. L'étude est ensuite approfondie en explorant le lien entre sélectivité fréquentielle et coût de calcul, avant d'introduire les outils de conception de filtres.

1 - Filtres numériques : interprétation et mise en œuvre

1.1 - Présentation

Un filtre numérique est défini par une relation entre un signal d'entrée $x[n]$ et un signal de sortie $y[n]$.

Dans le cas des filtres FIR (Finite Impulse Response), filtres à réponse impulsionnelle finie, dont les échantillons de sortie dépendent uniquement des échantillons d'entrée :

$$y[n]=\sum_{k=0}^{N} b_k \cdot x[n-k]$$

Chaque échantillon de sortie est obtenu comme une combinaison pondérée d'échantillons passés. Cette structure est simple à comprendre et garantit un comportement stable, au contraire des filtres IIR (Infinite Impulse Response), filtres à réponse impulsionnelle infinie dont les échantillons de sortie dépendent à la fois des échantillons d'entrée et des échantillons de sortie précédemment calculés et qui ne seront pas évoqués ici.

Dans GNU Radio, la relation de récurrence donnée ci-dessus se traduit directement par des blocs élémentaires. Cela permet de visualiser concrètement le rôle de chaque coefficient.

1.2 - Construction de filtres FIR simples

Moyenne glissante

Considérons le filtre :

$$y[n]=0{,}5 \cdot x[n]+0{,}5 \cdot x[n-1]$$

Que nous pouvons représenter par la structure suivante dans Gnuradio Companion :

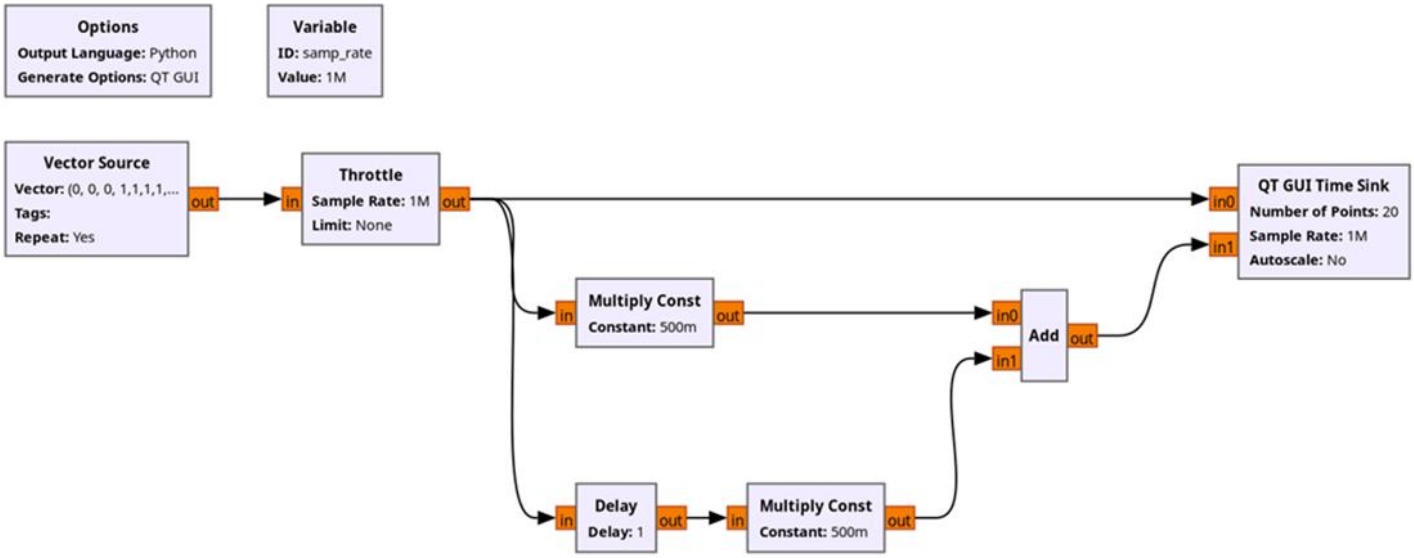

Figure 1 : Graphe GNU Radio Companion représentant un filtre à moyenne glissante soumis à un échelon en entrée

Tous les blocs utilisés sont configurés en type *Float* (ports de couleur orange), ce qui permet de manipuler directement des signaux réels. Le paramètre d'échantillonnage *samp_rate* est fixé à 1MHz, ce qui correspond à un intervalle de 1µs entre deux échantillons.

Le signal d'entrée du filtre est généré à l'aide d'un bloc *Vector Source*, produisant la séquence suivante :

(0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)

Ce signal correspond à un échelon discret, utilisé pour observer la réponse indicielle du filtre.

L'entrée et la sortie du filtre sont visualisées simultanément à l'aide d'un bloc *QT GUI Time Sink* configuré avec deux voies. Le déclenchement est réglé en mode automatique, avec un niveau de 0,1 et un délai de 2 µs, ce qui permet de stabiliser l'affichage autour du front de l'échelon.

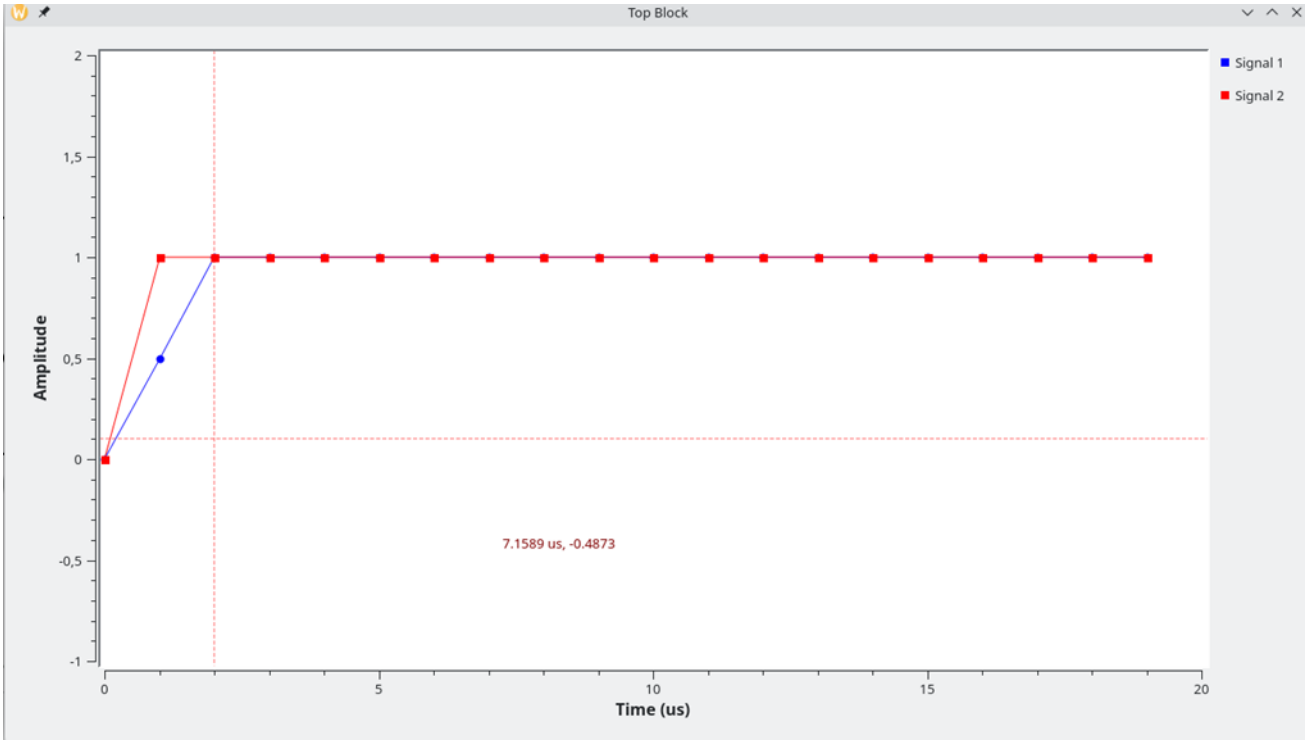

Figure 2 : Réponse indicielle du filtre à moyenne glissante

On observe sur la figure 2 que la sortie ne passe pas instantanément de 0 à 1, mais présente une transition progressive sur deux échantillons. Cette évolution est caractéristique du filtre utilisé, qui réalise une moyenne entre l'échantillon courant et le précédent. Cette observation permet de relier directement l'équation du filtre à son comportement temporel. Ce filtre réalise ainsi une moyenne entre deux échantillons successifs. Il atténue les variations rapides et agit donc comme un filtre passe-bas.

De la même façon, il est possible d'observer la réponse impulsionnelle du filtre en remplaçant l'échelon en entrée par un indice. Toutefois, une approche encore plus riche consiste à étudier son comportement lorsqu'il est excité par un signal sinusoïdal.

Pour cela, remplaçons le signal d'entrée précédent par un bloc *Signal Source* (Figure 3). Ce bloc permet de générer un signal périodique simple, ici un sinus, dont on peut régler facilement les paramètres principaux. L'amplitude est fixée à 1 afin de disposer d'une référence claire pour l'observation du gain du filtre, tandis que la fréquence devient le paramètre variable.

Afin de pouvoir faire varier cette fréquence de manière interactive, on introduit une variable nommée *freq*, associée à un bloc *QT GUI Range*. Ce bloc correspond à un curseur affiché dans la fenêtre de GNU Radio. En déplaçant ce curseur, on modifie en temps réel la fréquence du sinus généré, ce qui permet d'observer progressivement le comportement du filtre sur toute la bande de fréquences.

L'entrée et la sortie du filtre sont ensuite visualisées simultanément à l'aide d'un bloc *QT GUI Time Sink*. Pour améliorer la lisibilité des signaux, notamment lorsque les variations deviennent rapides ou que les déphasages sont faibles, chaque voie est suréchantillonnée à l'aide d'un bloc *Rational Resampler* configuré avec un facteur d'interpolation égal à 10. Cette opération consiste à insérer des échantillons intermédiaires entre les échantillons existants, sans modifier le contenu fréquentiel du signal, ce qui permet d'obtenir une représentation temporelle plus fine, notamment lorsque la fréquence des signaux se rapproche de la fréquence de Nyquist.

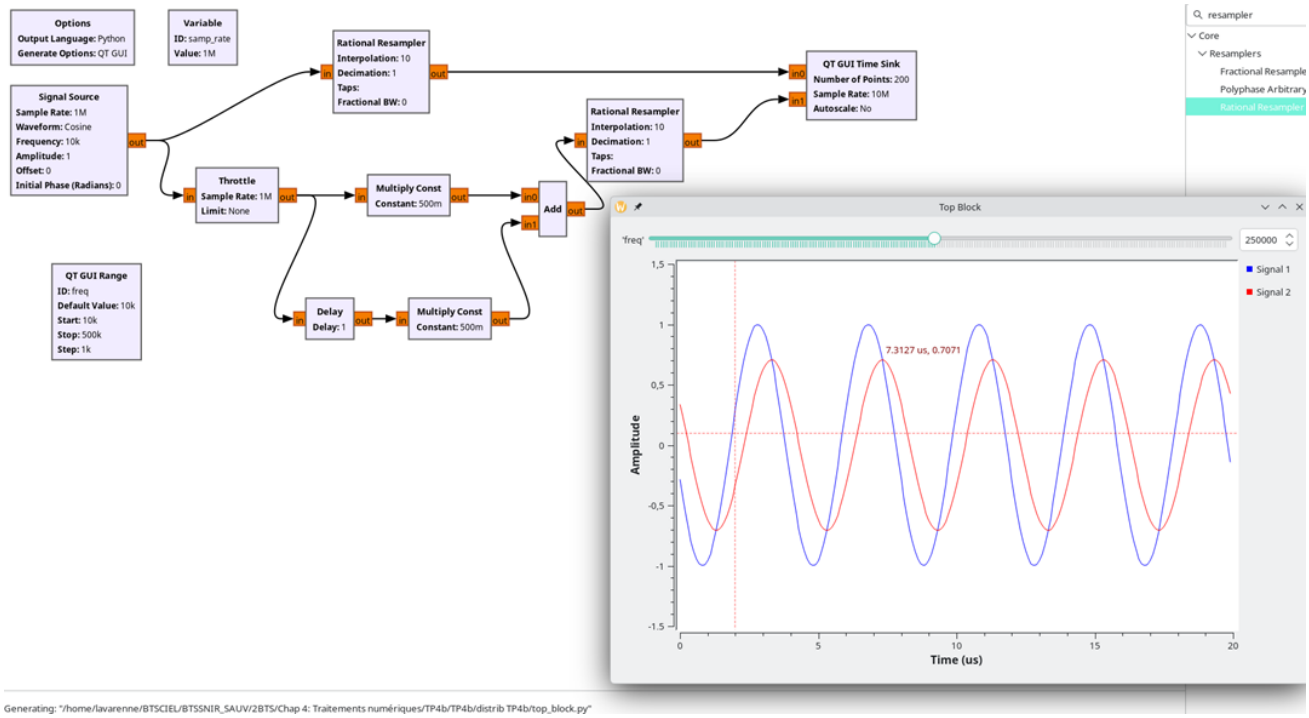

Figure 3 : Réponse du filtre à un signal sinusoïdal de fréquence 250 kHz. En bleu le signal d'entrée et en rouge le signal filtré en sortie

Comme le nombre d'échantillons par seconde est ainsi multiplié par 10, il est nécessaire d'adapter le paramètre *Sample Rate* du bloc *QT GUI Time Sink*, qui est ici réglé à 10 MHz. Cela garantit que l'axe temporel reste cohérent avec le signal affiché.

On peut alors faire varier progressivement la fréquence du sinus à l'aide du curseur. Pour les basses fréquences, le signal de sortie est quasiment identique au signal d'entrée. En revanche, lorsque la fréquence augmente, on observe une atténuation de l'amplitude ainsi qu'un déphasage entre les deux signaux. Lorsque l'amplitude du signal de sortie devient environ égale à 0,7 fois celle du signal d'entrée, on atteint la fréquence de coupure du filtre, caractérisée par une atténuation de -3 dB (Figure 3). En effet, pour le filtre défini par :

$$y[n]=0{,}5 \cdot x[n]+0{,}5 \cdot x[n-1]$$

La réponse fréquentielle vaut :

$$H(f)=0{,}5 \cdot \left(1+e^{-j2\pi f/f_e}\right)$$

Son module est :

$$|H(f)|=cos(\pi\, f/f_e)$$

La fréquence de coupure est définie par :

$$|H(f_c)|=\frac{1}{\sqrt{(2)}}$$

On obtient alors :

$$cos(\pi\, f_c/f_e)=\frac{1}{\sqrt{(2)}}$$

D'où :

$$f_c=\frac{f_e}{4}$$

On observe ainsi sur la Figure 3 que la fréquence pour laquelle l'amplitude du signal de sortie vaut environ 0,7 fois celle du signal d'entrée correspond bien à un quart de la fréquence d'échantillonnage, conformément à la théorie.

Pour finir, il est également possible de visualiser directement la réponse en fréquence du filtre en utilisant un bruit aléatoire en entrée et un bloc *QT GUI Frequency Sink* en sortie. Cette approche permet d'observer en une seule fois le comportement du filtre sur l'ensemble du spectre, sans avoir à faire varier la fréquence du signal d'entrée (Figure 4).

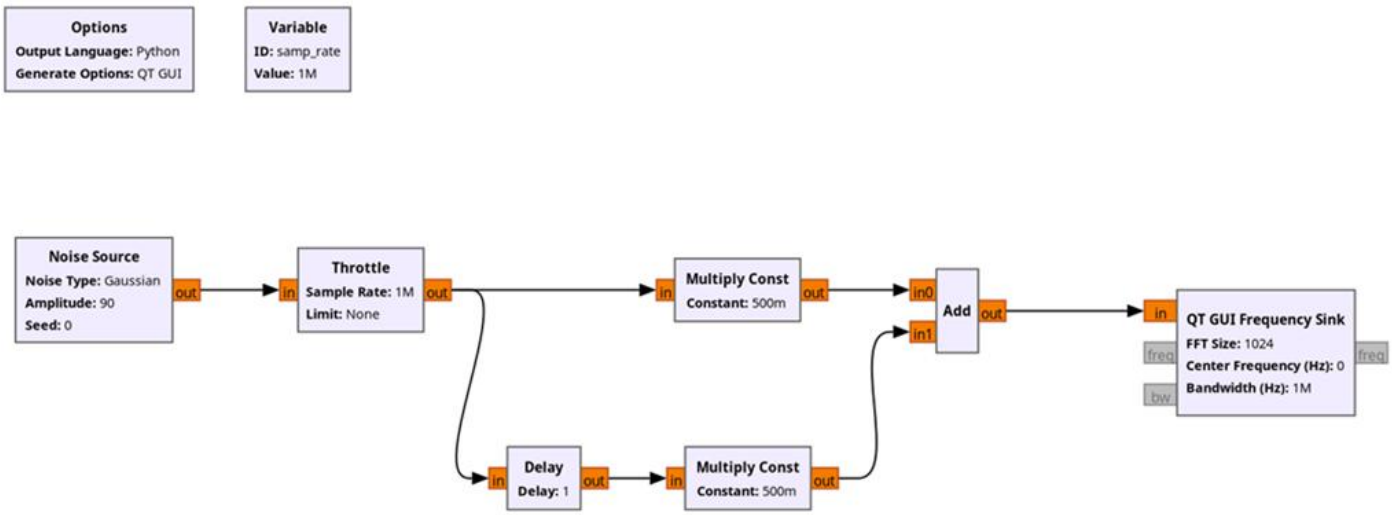

Figure 4 : Graphe GNU Radio Companion représentant un filtre à moyenne glissante soumis à un échelon en entrée

Le signal injecté est généré à l'aide d'un bloc *Noise Source*, configuré en bruit gaussien. Son amplitude est fixée à une valeur élevée (par exemple 90), ce qui ne modifie pas la forme de la réponse fréquentielle du filtre, mais permet d'obtenir un niveau de signal suffisamment élevé pour que le spectre affiché soit lisible.

En sortie, le bloc *QT GUI Frequency Sink* permet d'afficher le spectre du signal filtré. Il est important de régler correctement ses paramètres pour une interprétation claire. Le type de données est ici en *Float*, ce qui correspond à des signaux réels. Dans ce cas, le spectre affiché est symétrique, avec des fréquences positives et négatives. Il est donc pertinent d'activer l'affichage en mode « *Half* », afin de ne visualiser que la partie positive du spectre, ce qui simplifie la lecture.

Dans cette configuration, on observe clairement sur la Figure 5 que les basses fréquences sont transmises sans atténuation, tandis que les hautes fréquences sont progressivement atténuées. La transition entre ces deux zones correspond à la fréquence de coupure du filtre, que l'on retrouve autour d'un quart de la fréquence d'échantillonnage.

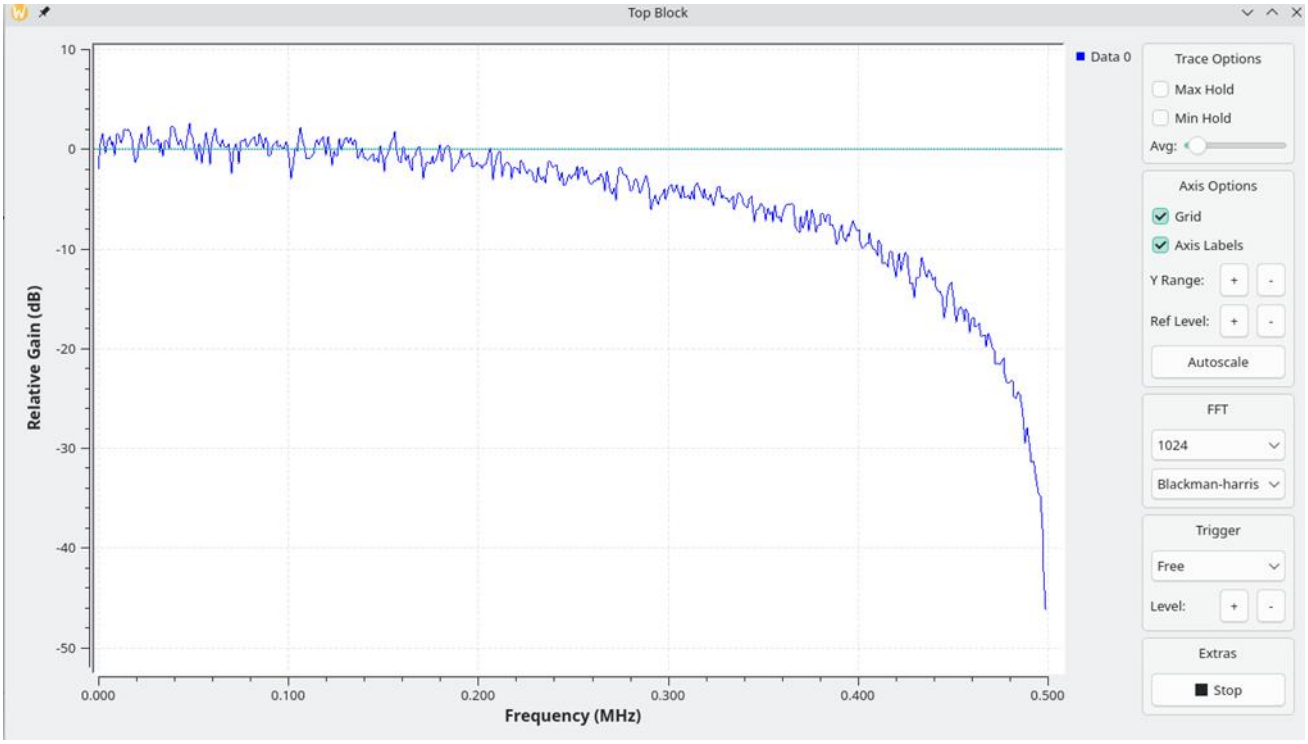

Figure 5 : Réponse en fréquence du filtre à moyenne glissante

1.3 - Filtre passe-bande

Soit le filtre caractérisé par l'équation suivante (Figure 6) :

$$y[n] = -0{,}2 \cdot x[n] + 0{,}05 \cdot x[n-1] + 0{,}7 \cdot x[n-2] + 0{,}05 \cdot x[n-3] - 0{,}2 \cdot x[n-4]$$

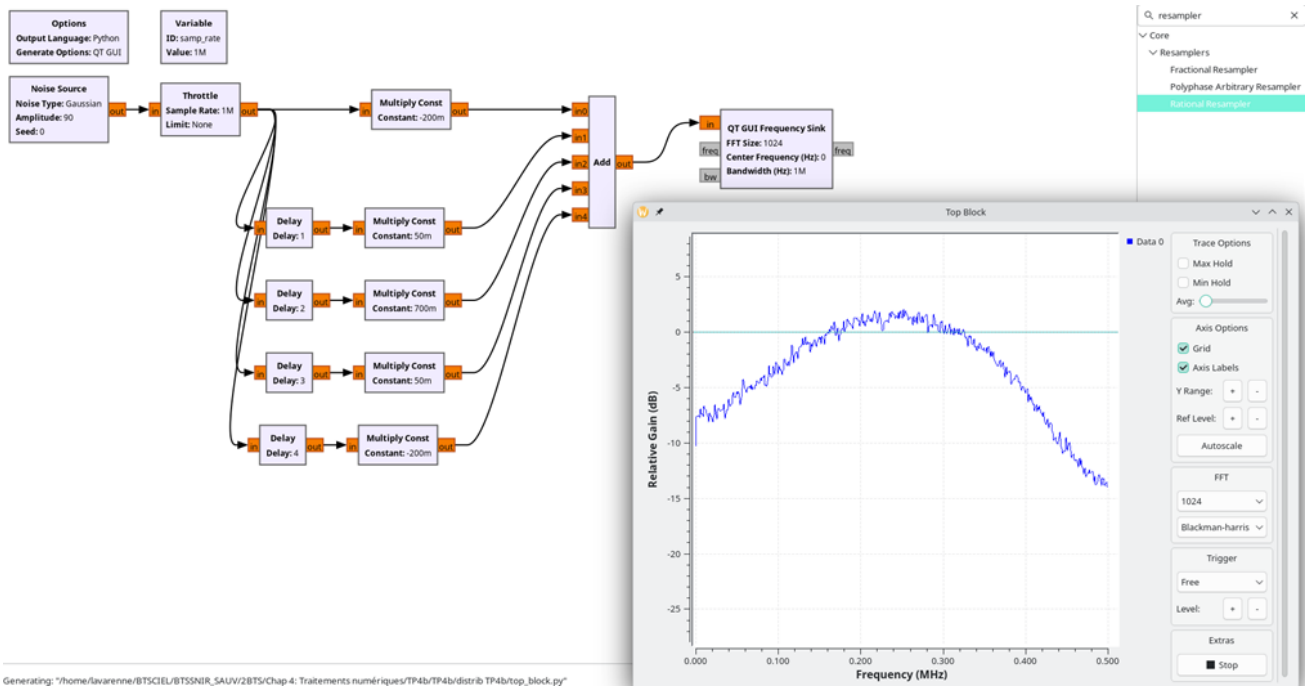

Figure 6 : Filtre passe bande à 5 coefficients

Contrairement aux exemples précédents, les coefficients ont ici été choisis de manière à privilégier une certaine bande de fréquences. La visualisation de la réponse fréquentielle, obtenue à l'aide d'un bruit en entrée et d'un bloc *QT GUI Frequency Sink*, met clairement en évidence un comportement de type passe-bande : les basses fréquences sont atténuées, le gain devient maximal autour d'une fréquence centrale, puis baisse à nouveau pour les hautes fréquences.

Cet exemple illustre un point important en filtrage numérique : la nature du filtre dépend directement des coefficients utilisés. Une simple modification de ces derniers permet de passer d'un filtre passe-bas à un filtre passe-haut, ou encore à un filtre passe-bande, sans changer la structure générale du montage. En modifiant en temps réel les coefficients (à l'aide de *QT GUI Range*), il devient possible d'observer immédiatement l'impact sur la réponse fréquentielle, ce qui permet de relier de manière concrète l'expression mathématique du filtre à son comportement spectral.

2 - Utilisation des blocs de filtrage dans GNU Radio

Après avoir étudié des filtres construits explicitement à partir de leur équation de récurrence, il est naturel de s'intéresser aux outils permettant de concevoir des filtres de manière plus directe. En pratique, on ne manipule généralement pas les coefficients un par un, mais on définit plutôt les caractéristiques fréquentielles souhaitées, telles que la fréquence de coupure ou la sélectivité.

GNU Radio propose pour cela des blocs de filtrage prêts à l'emploi, dont le plus courant est le bloc *Low Pass Filter*. Celui-ci permet de réaliser simplement un filtre passe-bas en spécifiant quelques paramètres physiques, les coefficients étant ensuite générés automatiquement par une fonction python.

2.1 - Paramétrage du bloc Low Pass Filter

Le bloc *Low Pass Filter* permet de réaliser un filtre passe-bas en spécifiant directement ses caractéristiques fréquentielles. Contrairement à l'approche précédente, il n'est plus nécessaire de définir explicitement les coefficients : ceux-ci sont calculés automatiquement à partir des paramètres fournis.

Les principaux paramètres à renseigner sont les suivants :

- La fréquence d'échantillonnage (**samp_rate**), identique à celle utilisée dans le reste de la chaîne de traitement ;
- La fréquence de coupure (**cutoff frequency**), qui définit la limite entre les fréquences transmises et celles qui seront atténuées ;
- La largeur de transition (**transition width**), qui correspond à la zone dans laquelle le gain du filtre passe progressivement de la bande passante à la bande coupée ;
- Le gain du filtre, généralement fixé à 1.

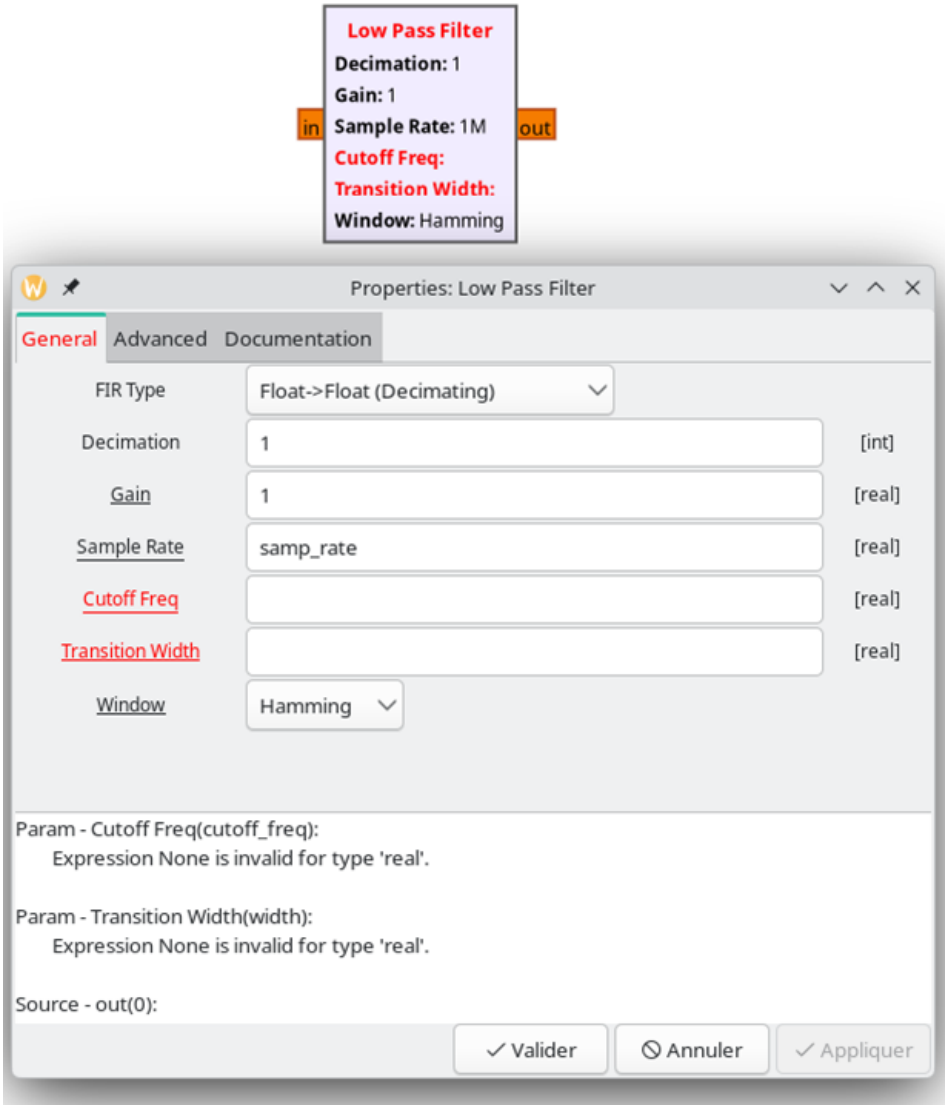

Figure 7 : Configuration du bloc Low Pass Filter

Cette manière de paramétrer le filtre est particulièrement simple et intuitive. On raisonne directement en termes de fréquences, ce qui facilite le lien avec le signal traité. Par exemple, pour un signal échantillonné à 1 MHz, une fréquence de coupure fixée à 100 kHz signifie que les composantes fréquentielles inférieures à cette valeur seront transmises, tandis que les composantes plus élevées seront progressivement atténuées. Une fois ces paramètres définis, GNU Radio génère automatiquement les coefficients du filtre FIR correspondant. L'utilisateur peut ainsi se concentrer sur le comportement du filtre plutôt que sur son implémentation.

2.2 - Observation de la réponse fréquentielle

Comme dans la partie précédente, la réponse fréquentielle du filtre peut être observée en injectant un bruit en entrée et en visualisant le spectre du signal en sortie à l'aide d'un bloc *QT GUI Frequency Sink*.

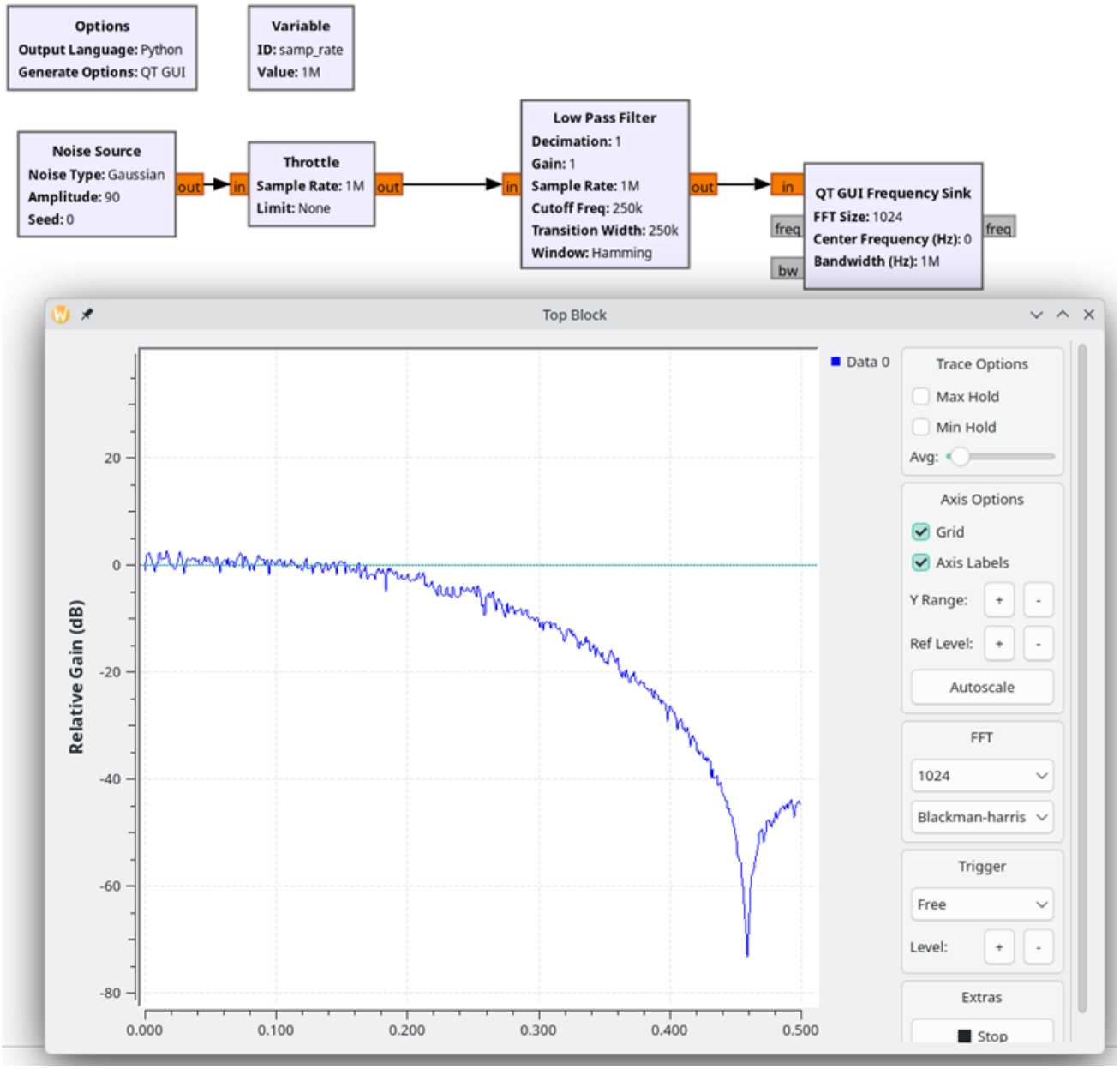

Figure 8 : Réponse en fréquence du Low Pass Filter avec cutoff = 250 kHz et Transition = 250 kHz

La réponse du filtre apparaît alors clairement. Les basses fréquences sont transmises, tandis que les hautes fréquences sont progressivement atténuées. La zone de transition entre ces deux comportements correspond à la fréquence de coupure définie dans le bloc *Low Pass Filter*.

Cette méthode permet d'observer en une seule fois l'ensemble de la réponse fréquentielle du filtre, sans avoir à faire varier la fréquence du signal d'entrée et constitue ainsi un outil simple et efficace pour analyser le comportement d'un filtre numérique.

2.3 - Influence de la largeur de transition

Un des paramètres importants du bloc *Low Pass Filter* est la largeur de transition (**transition width**). Elle définit la zone dans laquelle le gain du filtre passe progressivement de la bande passante à la bande atténuée.

Pour mettre en évidence son rôle, on peut associer ce paramètre à un bloc *QT GUI Range* afin de le faire varier en temps réel. L'observation de la réponse fréquentielle montre alors immédiatement l'effet de ce réglage.

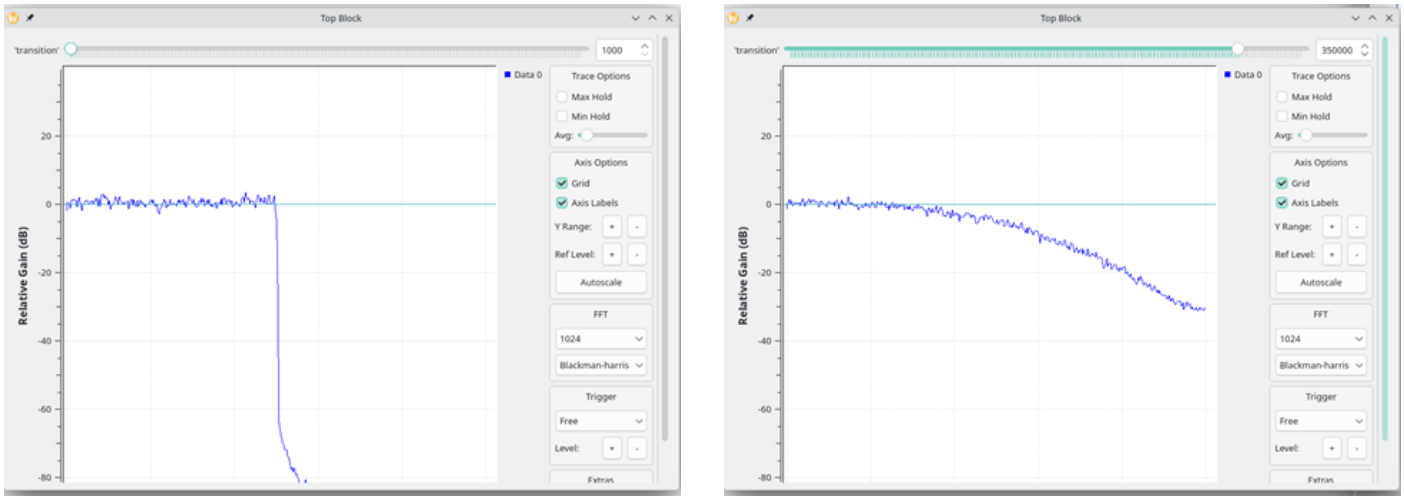

Figure 9 : Réponse en fréquence avec Transition = 1 kHz (à gauche) et Transition = 350 kHz (à droite)

Lorsque la largeur de transition est importante, la décroissance du gain est lente. Le filtre est peu sélectif : il laisse passer une partie des fréquences situées au-delà de la fréquence de coupure.

À l'inverse, lorsque la largeur de transition est réduite, la coupure devient beaucoup plus abrupte. Le filtre est alors plus sélectif, ce qui permet de mieux séparer les différentes composantes fréquentielles du signal. Cette amélioration a cependant un coût, car elle nécessite un filtre avec davantage de coefficients, comme nous allons le montrer dans le paragraphe suivant.

Cette observation met en évidence un compromis fondamental en traitement du signal numérique : il n'est pas possible d'obtenir simultanément une transition très abrupte et un filtre de faible complexité. Plus la transition est étroite, plus le nombre de coefficients nécessaires est important.

2.4 - Nombre de coefficients et coût de calcul

Le bloc *Low Pass Filter* masque volontairement la complexité du filtrage en générant automatiquement les coefficients nécessaires. Toutefois, ces coefficients existent bien, et leur nombre dépend directement des paramètres choisis, en particulier de la largeur de transition. Lorsque la transition est large, le filtre nécessite peu de coefficients. À l'inverse, lorsque la transition devient plus étroite, le nombre de coefficients augmente rapidement. Cette augmentation se traduit directement par une complexité de calcul plus importante.

En effet, pour chaque échantillon traité, le filtre doit effectuer une combinaison pondérée de l'ensemble de ses coefficients. Plus le filtre est long, plus le nombre d'opérations à réaliser est élevé. Dans une application en temps réel, ce point peut devenir non négligeable.

Cet aspect peut être mis en évidence en utilisant la fonction python de GNU Radio permettant de générer les coefficients du filtre :

firdes.low_pass(1, samp_rate, cutoff, transition)

Le nombre de coefficients générés dépend principalement de la largeur de transition. Lorsque celle-ci est large, le filtre est court et comporte peu de coefficients. En revanche, lorsque la transition devient très étroite, le nombre de coefficients augmente fortement.

Dans un shell python :

```
>>> from gnuradio.filter import firdes

>>> taps = firdes.low_pass(1, 1e6, 100e3, 100e3)

>>> print(taps)
[0.0020104029681533575, 0.001621020375750959, -1.0999144682209492e-18, -0.0044467085
97242832, -0.011685466393828392, -0.018134258687496185, -0.016773713752627373, 5.118
36987812244e-18, 0.035877108573913574, 0.08697697520256042, 0.14148786664009094, 0.1
8345333635807037, 0.19922685623168945, 0.18345333635807037, 0.14148786664009094, 0.0
8697697520256042, 0.035877108573913574, 5.11836987812244e-18, -0.016773713752627373,
-0.018134258687496185, -0.011685466393828392, -0.004446708597242832, -1.09991446822
09492e-18, 0.001621020375750959, 0.0020104029681533575]

>>> print(len(taps))
25
```

Ainsi, une bande de transition de 100 kHz nécessite la génération de 25 coefficients (la fonction len() en python renvoie le nombre d'éléments d'une liste). On peut diminuer la valeur de la bande de transition et observer l'effet sur le nombre de coefficient générés. Pour une bande de transition de 10 kHz :

```
>>> taps = firdes.low_pass(1, 1e6, 100e3, 10e3)

>>> print(len(taps))
241
```

Le nombre de coefficients est quasiment décuplé. On peut encore continuer, avec une bande de transition de 1 kHz :

```
>>> taps = firdes.low_pass(1, 1e6, 100e3, 1e3)

>>> print(len(taps))
2409
```

Nous constatons que la sélectivité fréquentielle d'un filtre non-récursif est directement liée à sa longueur. Plus la bande de transition est étroite, plus le nombre de coefficients nécessaires augmente.

Dans une implémentation temps réel, cela se traduit par une augmentation du nombre d'opérations à effectuer pour chaque échantillon. Le choix des paramètres du filtre ne relève donc pas uniquement de considérations théoriques, mais doit également tenir compte des contraintes de calcul.

3 - Conclusion

Cette première approche du filtrage numérique avec GNU Radio met en évidence un point essentiel : les concepts théoriques du traitement du signal prennent tout leur sens lorsqu'ils sont manipulés concrètement. La possibilité de visualiser simultanément les signaux temporels et fréquentiels permet d'établir un lien direct entre les équations de récurrence et leurs effets réels.

Les filtres FIR étudiés ici illustrent clairement comment le choix des coefficients influence le comportement du filtre, tandis que les blocs de conception intégrés à GNU Radio facilitent l'accès à des filtres plus complexes en se basant sur des paramètres physiques directement interprétables.

L'étude de la largeur de transition met également en évidence un compromis important entre sélectivité fréquentielle et coût de calcul, notion essentielle dans toute implémentation temps réel.