

TP4 : QPSK - Partie RF, filtre d'émission

Introduction

Nous consacrons deux séances à la QPSK. La première traite du modulateur, de la partie RF du démodulateur, de l'implémentation d'un filtre de mise en forme pour réduire la bande côté émetteur.

Dans la seconde, on abordera la partie bande de base du récepteur, et en particulier l'élimination du bruit et la récupération du timing symbole.

QPSK : rappels

Principe de base

En QPSK, les bits sont regroupés deux par deux ; chaque couple de bits détermine *la phase à l'origine des temps* du signal modulé. La forme d'onde obtenue pour un groupe de deux bits donné est appelée *symbole*.

Sur un temps symbole T_s , le signal modulé a pour expression :

$$s(t) = A \cos(\omega_c t + \varphi)$$

où φ est la phase à l'origine des temps, constante sur un temps symbole, et vaut $\pi/4$, ou $3\pi/4$, ou $5\pi/4$, ou $7\pi/4$.

Le signal $s(t)$ peut aussi être exprimé de la façon suivante, toujours sur le temps symbole :

$$s(t) = I(t) \cos(\omega_c t) - Q(t) \sin(\omega_c t) \quad (1)$$

avec

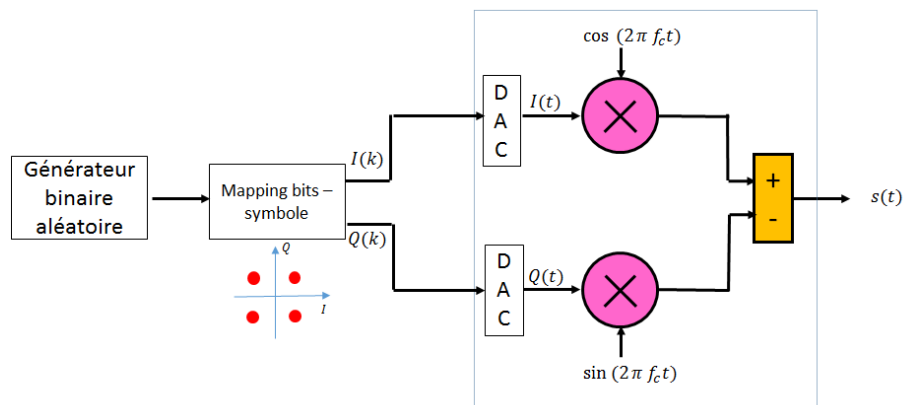
$$I(t) = A \cos(\varphi)$$

$$Q(t) = A \sin(\varphi)$$

$I(t)$ et $Q(t)$ sont des signaux *continus en temps* et, ici, *constants sur le temps symbole considéré* (car φ est constante).

Émetteur

De l'expression (1), on peut tirer un schéma de modulateur :



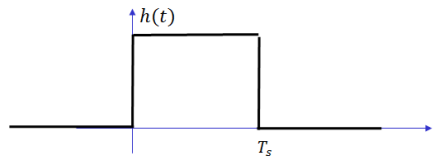
Sur ce schéma, il est très important de distinguer la partie *Bande de base*, à gauche avant la transposition en fréquence, c'est-à-dire la partie numérique et le DAC ; de la partie *RF*, dans laquelle principalement se fait la transposition vers la fréquence centrale du canal, par multiplication par la porteuse, et l'amplification (non représentée). Cette partie traite de signaux à des fréquences élevées (plusieurs GHz), hors de portée des systèmes numériques (imaginez que pour représenter une sinusoïde à 2 GHz avec 10 points par période, il faudrait 20 Gec/s...)

La partie encadrée du schéma ci-dessus se situera dans l'USRP.

Noter qu'un *modèle à temps continu* très souvent utilisé pour modéliser le DAC (en particulier pour appliquer la formule de Bennett) est :

$$D(t) = \sum_k D(k) \delta(t - k T_s) \xrightarrow{\quad} \boxed{h(t)} \xrightarrow{\quad} X(t)$$

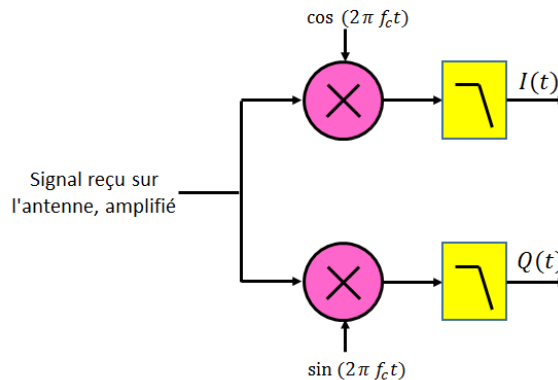
où $D(k) = I(k) + jQ(k)$, $X(t) = I(t) + jQ(t)$ et $h(t)$ est un filtre de réponse impulsionnelle rectangle :



Ce modèle fait qu'on parle couramment de *filtre rectangle* pour le schéma du modulateur de la page précédente. Retenir que "filtre rectangle" signifie " $I(t)$ et $Q(t)$ constants sur le temps symbole".

Récepteur

En réception, on utilise le schéma suivant pour récupérer $I(t)$ et $Q(t)$:




Noter que la partie bande de base (qui vise à obtenir les bits à partir de $I(t)$ et $Q(t)$) n'apparaît pas sur ce schéma. Nous verrons dans le prochain TP que si elle est simple dans une première approche, elle pose en réalité de vraies difficultés à cause des imperfections des oscillateurs et horloges.

1 Simulation

1.1 Émetteur

Nous allons commencer par simuler l'émetteur dans son intégralité, bande de base + RF. On rappelle pour une transmission réelle, le DAC et la partie RF se trouveront dans l'USRP.

1. Ouvrir le fichier `simu_QPSK_1`. Faire **CTRL+D** pour faire afficher les couleurs. Le modèle est déjà dessiné : vérifier qu'il correspond au schéma donné ci-dessus.
2. On commence par s'occuper de la partie bande de base à gauche. Double cliquer sur le bloc **Bernoulli Binary**, qui est le générateur de bits. Les bits sortent deux par deux. Le `sample time` renseigné ici est le temps bit ; quelle est sa relation avec le temps symbole ? Faire **CTRL+J** pour vérifier que cette partie du schéma est cadencée par la fréquence symbole.
3. Double-cliquer sur le bloc de Mapping, puis sur **View constellation**. On voit que chaque nombre complexe $I(k) + jQ(k)$ est en fait associé à un *entier* de 1 à 4. Chercher le paramètre à modifier pour que ce soit un couple de bits plutôt qu'un entier (ne pas passer à la suite tant que ce n'est pas fait), puis relever la constellation.
4. Observer les bits, $I(k)$ et $Q(k)$ sur le scope de gauche, et interpréter, à l'aide de la constellation.
5. On passe à présent à la partie RF (encadrée). Quels blocs génèrent $\cos(\omega_c t)$ et $\sin(\omega_c t)$? Que vaut la fréquence porteuse ? Qu'est-ce qui permet d'avoir un cosinus ou un sinus ?
6. Observer le signal modulé dans le scope de droite, mettre en évidence les changements de phase à l'origine des temps.
7. Décommenter le bloc **Spectrum analyser** à la sortie (clic droit > **Uncomment**). Faire passer le temps de simulation à `inf`, et quand le spectre s'affiche, zoomer sur la bande intéressante (autour de la fréquence porteuse bien sûr) en utilisant , pour garder environ la bande [0.6, 1.4] kHz environ.
Relever la largeur du lobe principal ; la comparer au débit de symboles. Relever la DSP à la fréquence porteuse, en dBm/Hz, puis en mW/Hz.

Comparer ces deux valeurs (largeur du lobe principal, valeur en 0) à ce que donne la formule théorique (Bennett) :

$$S_{XX}(f) = \frac{T_s}{2} \text{sinc}^2 [\pi(f + f_c)T_s] + \frac{T_s}{2} \text{sinc}^2 [\pi(f - f_c)T_s]$$

Rappels sur l'analyse spectrale : le traitement réalisé par le bloc **Spectrum analyser sans moyennage** calcule en fait la DSP d'un signal *déterministe*, ou alors *d'une seule réalisation* d'un processus aléatoire. Or la DSP d'une seule réalisation, aléatoire par définition, est aléatoire (ie en chaque f , on obtient une valeur aléatoire).

Pour un processus aléatoire, il faut normalement prendre *l'espérance* des DSP de plusieurs réalisations ; donc en pratique moyennner celles-ci¹. C'est pourquoi on a réglé le paramètre **Averages** à 100. Pour vous en convaincre, faire varier ce paramètre entre 1 et 500 par exemple, puis le remettre à 100.

Le nombre de points sur lequel on calcule la FFT, N , définit la *résolution*, c'est-à-dire l'écart fréquentiel entre deux points. Il est réglé par le **Window length**, avec **NFFT** sur **Auto**. Diminuer ce paramètre **Window Length** à 4096 ; la résolution vaut alors

$$\frac{f_e}{4096}$$

1. Le résultat de ce moyennage est toujours aléatoire, mais la variance autour de la valeur théorique peut être aussi faible que l'on veut si on moyenne suffisamment de DSP de réalisations – c'est la loi des grands nombres.

Ici f_e est fixé par le bloc **Zero-order Hold** situé juste avant l'analyseur et vaut 64 kHz ; soit une résolution de 16 Hz environ. Vous pouvez vérifier que dans un intervalle 100 Hz, il y a un peu plus de 6 points.

1.2 Le récepteur, partie RF

1. Ouvrir le fichier `simu_QPSK_2`. La partie RF du récepteur est ébauchée. (En pratique, cette partie serait dans l'USRP.)
2. Connecter les blocs et configurer les deux filtres passe-bas. Par défaut, ceux-ci sont mal configurés et ça ne fonctionnera pas si vous ne changez pas le paramètre **Passband edge frequency** ; attention, il faut entrer $2\pi \cdot f_p$ où f_p est la fréquence extrême de la bande passante. L'objectif est d'obtenir sur le scope un `I_demod` identique au `I` de l'émetteur (aux artefacts liés à la réponse du filtre près).

Indication : nous avons déjà parlé de ce filtre analogique dans le TP1 lors du passage en fréquence intermédiaire (son rôle est identique ici). Vous pouvez vous aider de l'analyseur de spectre qui suit en le connectant avant le filtre.

2 Émission réelle avec USRP

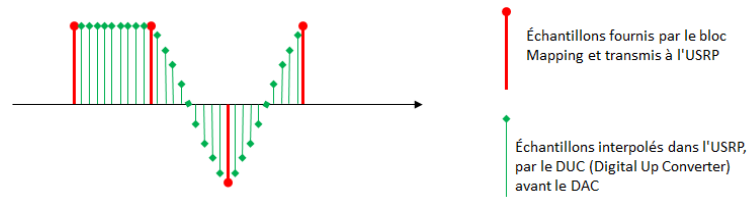
2.1 Première approche

1. Partir d'un fichier vierge, avec le template **Signal Processing**, le mode **accelerator**, un temps infini ; faire afficher les couleurs et les tailles des signaux.
2. On désire effectuer une transmission QPSK à 20 kb/s. Ajouter les blocs **Bernoulli Binary** pour générer les bits et **QPSK Modulator** pour le Mapping (les deux doivent être configurés par vos soins : ne pas laisser les paramètres par défaut ! En particulier, dans le générateur de bits, remplacer **Interpreted execution** par **Code generation** pour accélérer l'exécution). Insérer un bloc **SDRU Transmitter** précédé d'un **Buffer** pour implanter la partie RF dans l'USRP.
3. Pour l'instant, ne pas configurer le bloc, mais noter ci-dessous les réglages à effectuer pour que les paramètres **Master clock Rate** et **Interpolation** soient cohérents avec le sample rate à l'entrée de l'USRP :

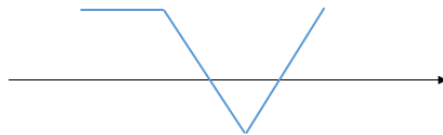
Master Clock Rate :

Interpolation :

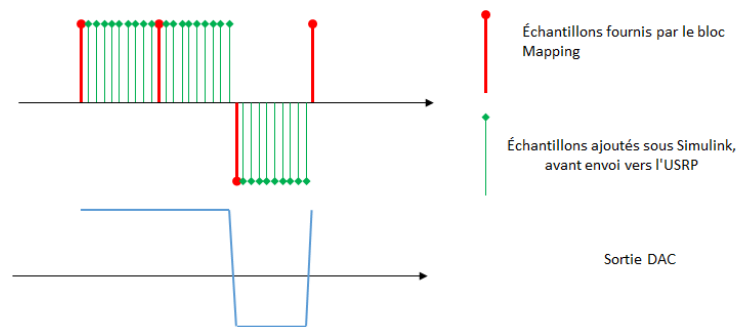
4. En fait, on ne peut pas procéder comme cela. Ceci est lié à l'étage d'interpolation précédant le DAC. On veut que $I(t)$ et $Q(t)$ soient constants sur un temps symbole (équation (1), allure "rectangle") ; or l'étage d'interpolation va insérer des échantillons entre deux échantillons consécutifs arrivant à l'entrée de l'USRP, non pas en maintenant constant le premier, mais précisément, *en interpolant*, donc en créant des échantillons de valeur intermédiaire entre les deux (on ne sait pas a priori s'il s'agit d'une interpolation linéaire ou autre, mais peu importe). L'allure de $I(t)$ et $Q(t)$ sera donc plutôt "triangulaire" au moment des changements de niveau, comme illustré ci-dessous : (sur le schéma, le facteur d'interpolation est 10) :



De sorte qu'à la sortie du DAC, on aura quelque chose comme :



Ce n'est pas l'allure souhaitée. C'est pourquoi il vaut mieux, dans Simulink, sur-échantillonner au préalable I et Q (ie augmenter le sample rate f_e), mais de sorte que les échantillons ajoutés soient *égaux* à l'échantillon qui précède, avant de les envoyer à l'interpolateur de l'USRP. Ce qui donnera, avec la même échelle :



Noter que s'il reste une interpolation effectuée par l'USRP, on aura toujours une pente au niveau du changement de niveau, mais l'effet sera négligeable.

Mettons que l'on souhaite augmenter le sample rate d'un facteur 10 sous Simulink. Pour cela, ajouter, directement à la sortie du bloc **QPSK Modulator Baseband** :

- un bloc **Upsample** de facteur 10, qui ajoute 9 échantillons nuls (de valeur 0) derrière chaque échantillon d'entrée. Dans **Rate options**, choisir **Allow multirate processing**;
- un bloc **Discrete FIR filter**, avec comme coefficients **ones(1,10)** (10 coefficients à 1).

La convolution de la sortie de l'**Upsample** par les échantillons du **FIR filter** va donner la sortie voulue.

Configurer le **SDRu Transmitter**, avec une fréquence centrale qui vaut, N étant votre numéro de binôme :

- si N vaut 1, 4, 7, 10 :

$$433 + N \times 0.15 \text{ MHz}$$

- si N vaut 2, 5, 8, 11 :

$$863 + N \times 0.4 \text{ MHz}$$

- si N vaut 3, 6, 9, 12 :

$$2480 + N \times 0.5 \text{ MHz}$$

5. Exécuter votre modèle. La Led rouge doit s'allumer et ne pas clignoter (sinon c'est que la transmission est discontinue).

A l'aide d'un analyseur de spectre, observer le spectre de votre émetteur QPSK. Est-ce bien le spectre prévu ?

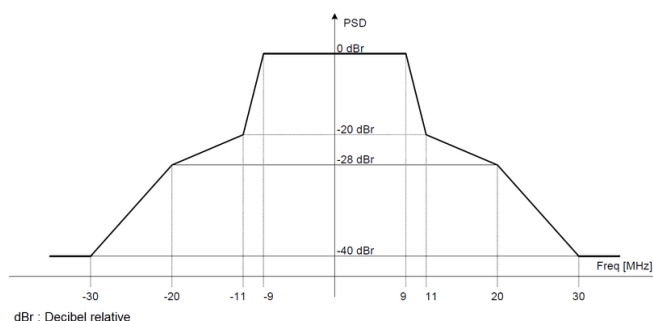
Le problème du filtre rectangle, ce sont les lobes secondaires du spectre. Ces lobes ne sont pas vraiment utiles car la majorité de la puissance se situe dans le lobe principal ; pour autant, on ne peut pas émettre un autre signal là où se trouvent les lobes secondaires, car ce dernier serait perturbé.

Il faut comprendre que le spectre électromagnétique est complètement occupé. C'est une ressource rare ! L'État peut, pour le déploiement des réseaux mobiles par exemple, libérer des bandes ; le prix que payent les opérateurs pour avoir le droit de les exploiter est une bonne indication de leur valeur. En 4G, dans la bande des 700 MHz, les opérateurs ont payé près de 1 Milliard d'euros pour 10 MHz de bande.

(www.lesechos.fr/2015/11/les-encheres-pour-les-frequence-4g-sont-bel-et-bien-terminees-281970)

A ce prix là, on ne peut se permettre de gâcher des morceaux de spectre à cause de lobes secondaires. Tous les standards, qu'il s'agisse de réseaux mobiles, Wifi, ADSL, etc, spécifient un *spectral mask* qui donne l'atténuation minimum du signal par rapport à son maximum (on note l'atténuation en dBr, ce qui signifie dB relativement au maximum), à certaines distances de la fréquence porteuse du canal.

A titre d'exemple, le masque spectral du 802.11 a (Wifi), où le canal a une largeur de 20 MHz :



On voit qu'à 9 MHz du centre (sur le bord droit du canal), le signal n'a pas besoin d'être atténué ; en revanche, à 11 MHz (sur le bord gauche du canal voisin), il doit être atténué de 20 dB (par rapport au maximum) ; et à 20 MHz (centre du canal voisin) de 30 dB.

BONUS : Imaginons que vous disposiez d'un canal de 20 kHz. Pour quantifier le débordement de votre émetteur, on peut mesurer l'*ACPR* (Adjacent Channel Power Ratio), rapport entre la puissance que vous émettez dans un canal adjacent et de la puissance émise dans votre propre canal (vous pouvez vous aider de l'Annexe 4 du TP1, et si nécessaire du manuel de l'analyseur – celui-ci est disponible en version papier dans la salle ou sur Moodle, section TP1). Utiliser pour la mesure un span de 60 kHz au moins, et dans les paramètres de la mesure, 20 kHz comme largeur de canal, et également 20 kHz comme espacement entre porteuses.

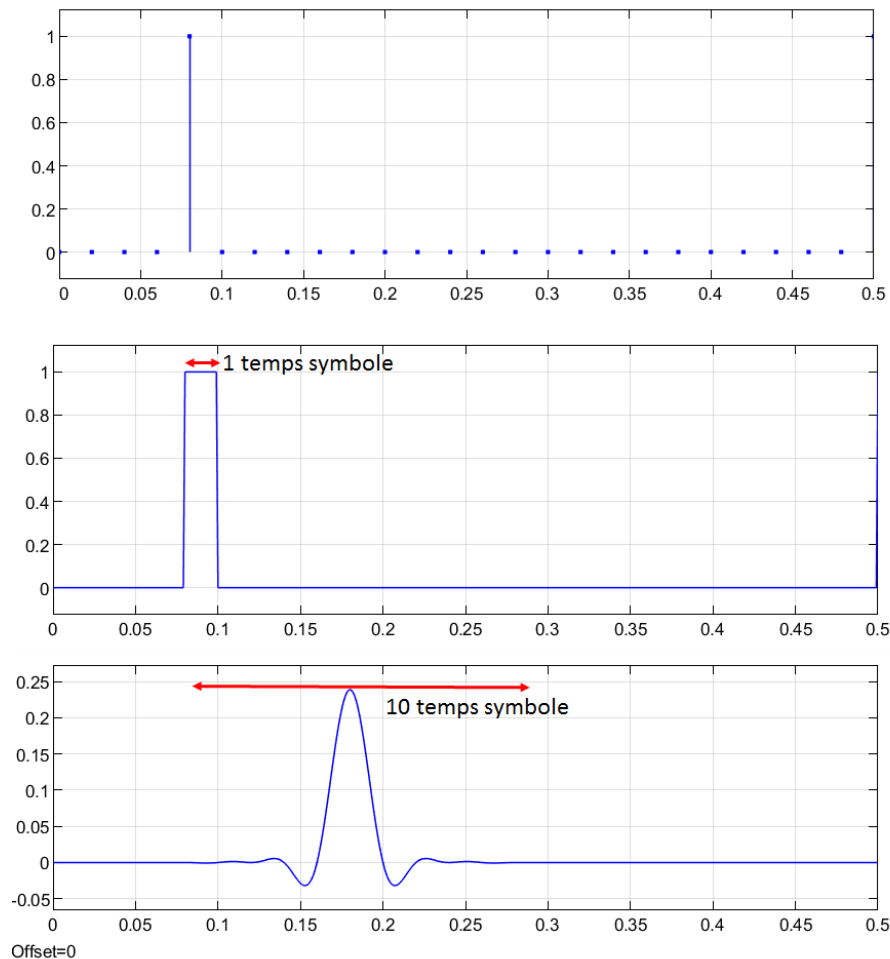
Pour éviter qu'un émetteur ne "bave" autant sur les canaux voisins, on utilise un *filtre de mise en forme*, qui réduira la bande occupée au strict minimum.

2.2 Le filtre d'émission (ou de mise en forme)

Le filtre en cosinus surélevé (raised cosine filter) est très souvent utilisé car il permet d'éviter l'Interférence Entre Symboles (IES ou ISI en anglais). Nous reviendrons sur cet aspect lors du TP5.

Le facteur de roll-off, souvent noté α , est un paramètre du filtre qui détermine dans le domaine fréquentiel la raideur de coupure (plus le roll-off est faible, plus la pente est raide) ; et dans le domaine temporel, le temps de montée (plus le roll off est faible, plus le temps de montée est important – plus le signal est déformé).

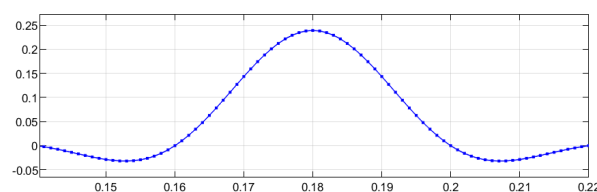
Pour implanter ce filtre, on cherche à générer sous Simulink – donc avant envoi vers l'USRP – un signal discret ayant l'allure désirée.



On a représenté ci-dessus, avec toujours 50 symboles/s, soit un temps symbole de 0.02 s :

- un échantillon du signal discret $I(k)$ ou $Q(k)$ ($I(k)$ et $Q(k)$ contiennent un seul échantillon par temps symbole, défini par le couple de bits) ;
- la sortie du DAC pour un filtre rectangle ;
- la réponse pour un filtre en cos surélevé.

Si un signal discret doit représenter cette sortie, il doit nécessairement comporter *plusieurs échantillons par temps symbole*. Ci-dessous, un zoom sur 4 temps symboles, en mettant en évidence les échantillons (ici on a 10 échantillons par temps symbole) :



Noter que si l'on avait 1 seul échantillon par temps symbole, aux instants 0.14, 0.16, 0.18, 0.20, 0.22, on aurait des échantillons nuls sauf en 0.18, donc pas du tout l'allure de la réponse souhaitée ! Le rapport entre les sample rates à la sortie et à l'entrée du filtre est appelé *facteur de sur-échantillonnage* ou *OverSampling Ratio* (OSR).

Pour générer le signal discret qu'on enverra à l'USRP, la méthode est la même que précédemment : une étape d'**Upsample** qui consiste à insérer des 0 entre les échantillons de $I(k)$ et $Q(k)$ (on insère OSR-1 échantillons nuls) ; et une étape de filtrage numérique par un filtre FIR, dont les coefficients sont obtenus en échantillonnant la réponse impulsionnelle théorique.

1. Modifier le taux de suréchantillonnage de votre bloc **Upsample**, pour choisir un OSR compris entre 4 et 10 (soit 4 à 10 échantillons par temps symbole).
2. Pour obtenir les coefficients du filtre FIR, remplacer les coefficients `ones(1,10)` par `rcosdesign(α , Span, OSR, Shape)`, où α est le facteur de roll-off ; OSR le facteur de sur-échantillonnage ; Shape une chaîne de caractères parmi '`normal`' et '`sqr`'. Pour nous, pour le moment, c'est '`normal`' qu'il faut choisir. (Nous y reviendrons dans le TP5).

Noter que la réponse impulsionnelle théorique du filtre a une durée infinie ; elle doit être *tronquée* à un certain nombre de temps symbole (ci-dessus par exemple, elle est tronquée pour ne durer que 10 temps symbole). Le paramètre **Span** est le nombre de temps-symbole sur lequel s'étale la réponse impulsionnelle tronquée (forcément pair, typiquement 6 à 12).

3. Avant de tester le filtre, on peut observer sa réponse théorique sous Matlab. Pour cela, taper dans la fenêtre de commandes Matlab (en remplaçant les paramètres par les valeurs numériques choisies) :

```
my_filter=rcosdesign( $\alpha$ , Span, OSR, Shape)
fvtool(my_filter)
```

Vous devez voir apparaître la réponse fréquentielle. Pour accéder à la réponse impulsionnelle, cliquer sur 

Sur combien d'échantillons s'étale cette réponse ? Pouvait-on déduire cela des paramètres du filtre ? Au bout de combien d'échantillons la réponse à une impulsion (un Dirac) atteint-elle son maximum ? Quel est le retard occasionné par le filtre, en temps symbole ?

4. Attention ! Le sample rate à l'entrée de l'USRP a changé si vous avez changé le paramètre de l'**Upsample**. Modifier les paramètres **Master Clock Rate** et **Interpolation** du bloc **SDRU Transmitter** en conséquence. Puis exécuter votre modèle, et à l'aide de l'analyseur, observer le spectre de votre signal. Vérifier que l'encombrement spectral a diminué, et mesurer à l'aide de l'analyseur l'**Occupied Bandwidth** (OBW), bande dans laquelle réside 99% de la puissance (voir l'Annexe 4 du TP1).

On utilise souvent comme bande théorique : $D_s \times (1 + \alpha)$, où D_s est le débit de symboles et α est le rolloff. Vérifier que les deux ne sont pas identiques (cette bande n'est pas l'Occupied Bandwidth mais plutôt la bande à 3 dB.)

Changer α et observer son effet. Si vous l'avez mesuré dans la partie 2.1, mesurer de nouveau l'ACPR.