TD GNU Radio

Sujet de E. Monier et J-P. Barbot

0. Introduction et objectifs

GNU Radio est une boîte à outils open-source pour la construction de radios logicielles. Contrairement aux radios classiques, ce n'est pas un hardware fixe qui définit les opérations appliquées aux signaux, mais un software fortement modulable. GNU Radio est un environnement de traitement du signal qui englobe des centaines de blocs de traitement du signal et de communications numériques, tous développés en C++ ou Python. Il permet d'émettre et de recevoir des signaux échantillonnés à l'aide de cartes telles que celles vendues par Ettus ou NI (carte USRP) ou de recevoir des signaux démodulés à l'aide de clefs USB Tuner TNT utilisant le composant RTL2832U+R820T. Le principal objectif de cette partie est de comprendre l'environnement GNU Radio, d'étudier le fonctionnement des blocs essentiels ainsi que leur action sur les signaux (filtres, sources, modulateurs, etc).

1. Familiarisation avec les différents blocs

Le menu de droite liste l'ensemble des blocs de traitement disponibles dans GNU Radio. Pour rechercher un bloc particulier, utiliser la loupe et tapez les lettres d'un nom de bloc. Par exemple source. Dans un premier lieu, un bloc Signal Source (type cosinus) sera mis en oeuvre et nous observerons à l'aide d'un WX Scope Sink les différents signaux qu'on peut obtenir en modifiant le type du signal généré. Remarquez que les environnements graphiques du type Scope peuvent être générés à l'aide de deux générateurs de GUI (Graphic User Interface) : WX ou QT. Vous ne pouvez utiliser qu'un type à la fois, celui-ci étant défini dans le bloc principal Options (c'est la première erreur que vous risquez de rencontrer).

Manipulation 1 Réalisez le graphique décrit ci-dessus. Pour relier deux blocs, cliquez sur le port de sortie de l'un puis sur le port d'entrée du suivant. Pour effacer un lien ou un bloc, sélectionnez-le et appuyez sur la touche Suppr. Pour démarrer, cliquez sur la flèche verte en haut à gauche. Un script Python est généré et exécuté.

a) En observant l'occupation du CPU, que remarquez-vous ?

b) Mettez en oeuvre un bloc **Throttle** pour y remédier.

Manipulation 2 Pourquoi a-t-on deux signaux différents dans l'affichage de **WX Scope Sink** ? Que représentent-ils ? Vous pouvez changer le type d'affichage en utilisant l'onglet **Marker**.

Manipulation 3 Modifiez le type de sortie du bloc **Signal Source** de manière à avoir une sortie réelle. Vous devrez aussi modifier le type du bloc **WX Scope Sink**.

Manipulation 4 Essayez les autres types de signaux proposés par **Signal Source**.

Manipulation 5 Maintenant, nous allons nous intéresser au bloc **Random Source** et visualiser les signaux générés.

a) On obtient une erreur. Quelle en est la cause ?

b) Pour résoudre le problème, il faut utiliser un bloc de conversion de type tel que **Short To Float**.

Manipulation 6 Modifiez le bloc de sortie de manière à afficher l'histogramme des échantillons générés. Pour cela, vous pourrez utiliser le bloc **WX GUI Histo Sink**.

Manipulation 7 Modifiez l'amplitude du signal généré par le **Random Source**. Que constatez-vous ?

2. Etude dans le domaine spectral

Avant de poursuivre, il faut comprendre la notion de variable. Si vous observez le bloc **Variable**, vous observez qu'il demande deux arguments : un ID et une valeur. Ce bloc permet de créer une variable ayant pour nom l'ID renseignée dans le bloc et lui assigne une valeur. Cette variable peut alors être appelée pour renseigner un paramètre dans un bloc. Observez par exemple le bloc **Variable** définissant la fréquence d'échantillonnage à travers le nom **samp_rate** et lui assignant par défaut la valeur 32kHz. Observez à présent les blocs que vous avez utilisé auparavant et retrouvez l'utilisation de cette variable. A présent, ce type de définition doit être utilisée car elle présente un intérêt certain : *elle permet de changer un paramètre du graphe sans avoir à redéfinir tous les paramètres de tous les blocs*.

Dans cette partie, il s'agit de générer un sinus de fréquence 1 kHz et d'amplitude 2 V. Rajoutez le bloc **Throttle** se trouvant dans le menu de droite et ajustez le Sample Rate à 32 kHz. Ensuite, rajoutez le bloc **WX GUI FFT Sink**.

Manipulation 8 Visualisez le signal temporel et son spectre. Commenter. Afin de rendre l'affichage plus compact, vous utiliserez le bloc **WX GUI Notebook** (afin de connaître son fonctionnement, reportez-vous à la documentation).

Nous avons vu jusqu'à présent le fonctionnement des blocs Variable. Un bloc semblable est le bloc **WX GUI Slider**. De même que pour le bloc **Variable**, il définit une variable et lui assigne une valeur. La différence est que la valeur associée à la variable peut être modifiée de manière dynamique dans la fenêtre GUI - celle-ci pouvant varier uniquement dans un intervalle définit dans le bloc. Une valeur par défaut est aussi renseignée. De même que pour le bloc **Variable**, la variable peut également être appelée dans d'autres blocs.

Manipulation 9 Ajoutez un bloc **WX GUI Slider** afin de changer la fréquence de la sinusoïde. Dans ce Slider, donnez un nom à la variable (par exemple freq), une valeur min (100) et max (30000), ainsi qu'une valeur par défaut. Dans le bloc Signal Source, à la place de la fréquence 1000, mettez la variable freq. Observez ce que vous obtenez et commentez.

Manipulation 10 Que se passe-t-il lorsque la fréquence de la sinusoïde est trop élevée ? Expliquez ce phénomène.

3. Modulation angulaire

On souhaite réaliser une modulation d'amplitude analogique à deux niveaux (i.e. une 2-ASK). Pour cela, on multiplie un signal carré (fréquence de 200 Hz, d'amplitude crête à crête de 2V et de moyenne nulle) par un signal sinusoïdal (fréquence de 4 kHz, amplitude 2V). On a vu en cours que cela revient aussi à faire une 2-PSK ou BPSK.

Manipulation 11 Créez un nouveau fichier et placez les blocs **Signal Source**, **Throttle**, **Multiply**, **WX GUI Scope Sink** et **WX GUI FFT Sink** afin d'observer le signal échantillonné et le spectre correspondant. La fréquence d'échantillonnage sera par exemple de 32 kHz.

Manipulation 12 Détaillez les graphes obtenus. Que constatez-vous ?

Manipulation 13 Rajoutez le bloc **WX GUI Waterfall Sink**. Qu'obtient-on ? Expliquez.

Manipulation 14 Maintenant, ajoutez une source de bruit en utilisant les blocs **Noise Source** et **Add**. Faites varier le niveau de bruit à l'aide d'un **Slider** et observez les différents résultats, conclure. Expliquez précisément le fonctionnement du schéma réalisé précédemment.

Manipulation 15 En vous basant sur vos connaissances, démodulez le signal afin de recevoir le signal rectangulaire. Pour cela, vous pourrez utiliser la même porteuse que celle qui est utilisée à la source. Vous aurez également besoin des blocs **Multiply** et **Low Pass Filter**.

4. Modulations numériques

4.1 Rappel : chaîne de transmission numérique

Afin de bien comprendre la suite du tp, il est important de bien concevoir les rappels suivants.

4.1.1 L'émetteur

L'émetteur numérique est détenteur de l'information numérique à transmettre. Celle-ci se présente sous la forme d'un train binaire α_k - celui-ci ayant été compressé et codé - ne pouvant prendre que 2 valeurs : 1 ou 0.

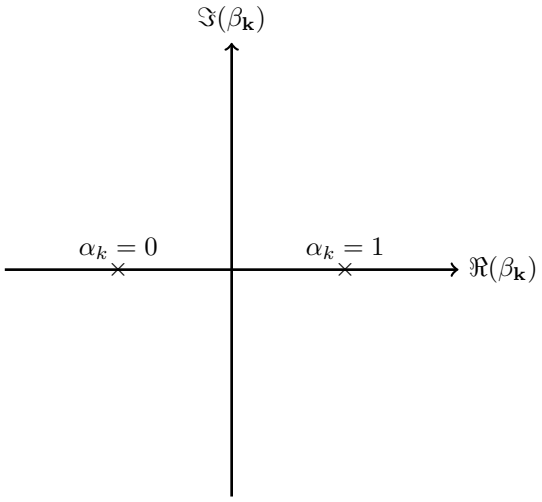
Ce train binaire est transmis en suivant les étapes définies ci-dessous.

Codage du message : Le train binaire α_k est "découpé" en paquets de n bits. A chaque paquet possible est associé un caractère réel ou complexe dans un dictionnaire χ. D'un message logique, on passe donc à une suite de caractères β_k .

Exemple 1 : ASK-2 ou BPSK. Le cas le plus simple est de découper le train α_k bit par bit. Selon la valeur de α_k , on en déduit automatiquement la valeur de β_k en suivant la règle suivante :

$$\beta_k = \begin{cases} -1 & \text{si } \alpha_k = 0 \\ +1 & \text{si } \alpha_k = 1 \end{cases}$$

Cette association peut être représentée à l'aide de la figure suivante.

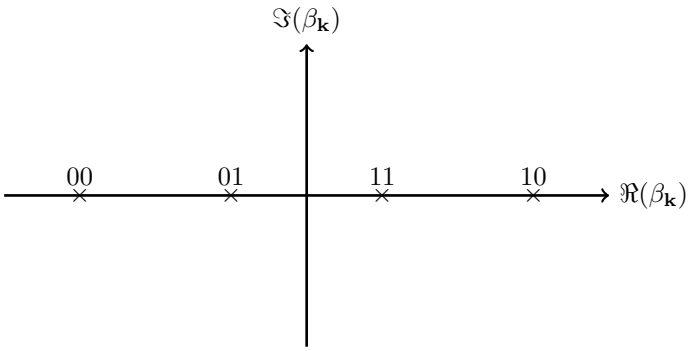


Ainsi, le dictionnaire est l'ensemble $\chi = \{-1,\ +1\}$.

Exemple 2 : ASK-4. Un autre exemple consiste à découper le train α_k par paquet de 2 bits. La fréquence de "sortie" des échantillons β_k est alors la moitié de la fréquence "d'entrée" des α_k. Selon la valeur d'un paquet $(\alpha_{2k-1}, \alpha_{2k})$, on en déduit automatiquement la valeur de β_k en suivant la règle suivante :

$$\beta_k = \begin{cases} -3 & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 00 \\ -1 & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 01 \\ +1 & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 11 \\ +3 & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 10 \end{cases}$$

Cette association est représentée à l'aide de cette figure.

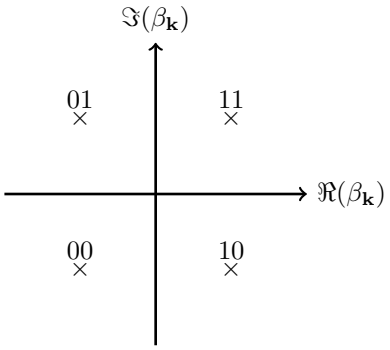


Ainsi, le dictionnaire est l'ensemble $\chi = \{-3,\ -1,\ +1,\ +3\}$.

Exemple 3 : PSK-4. Un dernier exemple découpe toujours le train α_k par paquet de 2 bits. La fréquence de "sortie" des échantillons β_k est alors la moitié de la fréquence "d'entrée" des α_k. Selon la valeur d'un paquet $(\alpha_{2k-1}, \alpha_{2k})$, on en déduit automatiquement la valeur de β_k en suivant la règle suivante :

$$\beta_k = \begin{cases} -1-i & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 00 \\ -1+i & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 01 \\ +1+i & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 11 \\ +1-i & \text{si } (\alpha_{2k-1}, \alpha_{2k}) = 10 \end{cases}$$

Cette association est représentée à l'aide de cette figure.

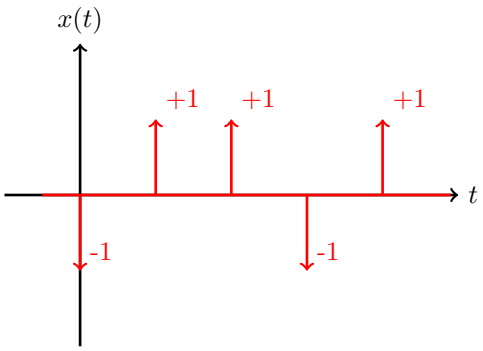


Ainsi, le dictionnaire est l'ensemble $\chi = \{-1-i,\ -1+i,\ +1+i,\ +1-i\}$.

Mise en forme du signal et création du signal en bande de base . Pour l'heure, nous ne disposons que d'une suite de caractères qu'il faut transformer en signal analogique qu'il va falloir transmettre. Pour cela, définissons le signal $x(t)$ comme suit :

$$x(t) = \sum_{k=0}^{N} \beta_k \delta(t - kT)$$

où T est la durée symbole et où N est le nombre d'échantillons β_k. La représentation de $x(t)$ en considérant la suite $\beta = \{-1,\ 1,\ 1,\ -1,\ 1\}$ est représentée ci-dessous



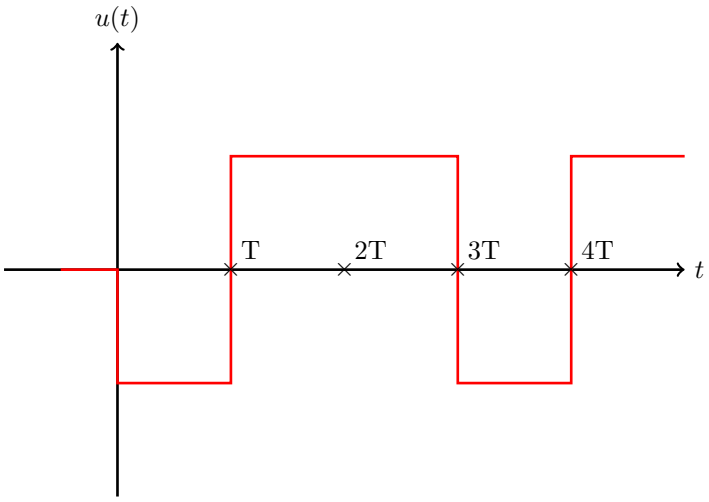
Afin d'obtenir le signal en bande de base, il faut donner à ce signal une réalité temporelle. Pour cela, il faut utiliser un filtre d'émission qui aura plusieurs buts :

— filtrer les hautes fréquences du signal $x(t)$ afin de permettre aux utilisateurs de ne pas utiliser une bande passante trop large,

— donner certaines propriétés au signal.

Le signal $u(t)$ sortant du filtre est appelé **signal en bande de base** et est égal à $(x * h)(t)$ où $h(t)$ est la réponse impulsionelle du filtre d'émission. On voit alors se dessiner des éléments se répétant et ayant la forme de la réponse impulsionelle h qu'on appelle symboles.

Le premier filtre qu'on puisse considérer est celui ayant comme réponse impulsionelle la porte de durée T. Cela permet de donner l'allure suivante au signal.



Bien que cette impulsion soit intéressante pour la détection des caractères, sa transformée de Fourier - un sinus cardinal - présente de larges lobes très peu amortis. Par conséquent, très souvent, ce sont des impulsions plus complexes qui sont choisies, comme celle de Nyquist.

Modulation . Une fois le signal en bande de base réalisé, il s'agit de transposer ce signal dans les hautes fréquences afin de permettre la transmission. Pour cela, on multiplie le signal en bande de base (modulante) avec sa porteuse $p(t) = Ae^{j2\pi f_p}$. Le signal émis est donc $s(t) = (u \cdot p)(t)$.

4.1.2 Le canal de transmission

Le modèle de canal de transmission que nous utiliserons ici comprend actions sur le signal émis $s(t)$:

— le canal atténue le signal,

— du fait de la distance entre l'émetteur et le récepteur, il retarde également le signal,

— il ajoute enfin du bruit.

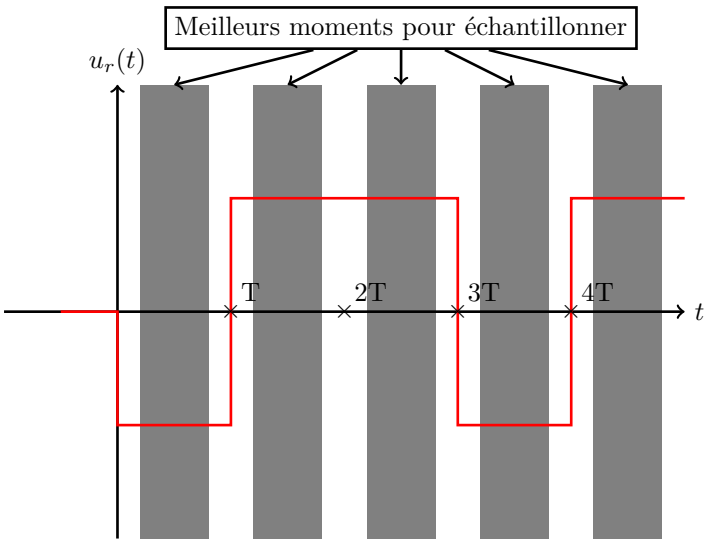
Ainsi, le signal reçu est

$$s_r(t) = K \cdot s(t - \tau) + b(t)$$

4.1.3 Le récepteur

La démodulation permet d'abord de retrouver le signal en bande de base en le ramenant au voisinage de 0. La démodulation est réalisée en multipliant le signal reçu par la fréquence porteuse - qu'il faut récupérer - et en ne conservant que le signal en bande de base à l'aide d'un filtre passe-bas. Le signal en bande de base retrouvé est noté $u_r(t)$.

Retrouver les caractères . Une fois le signal en bande de base retrouvé, il s'agit de retrouver la suite des caractères en échantillonnant le signal $u_r(t)$ aux bon instants. La figure ci-dessous fait apparaître cette problématique.



Retrouver les meilleurs moments pour échantillonner passe par le besoin de retrouver la fréquence d'horloge. Classiquement, le meilleur moment pour échantillonner peut être retrouvé en corrélant $u_r(t)$ par l'impulsion $h(t)$ et en déterminant les pics de la fonction d'autocorrélation.

Une fois cette étape réalisée, une suite de symboles $\beta_{r,k}$ est obtenu. L'affichage de ces échantillons dans le plan complexe correspond à la **constellation**.

Retrouver le message initial . Pour récupérer l'information transmise, l'inverse de l'opération de codage est effectuée. Par exemple, en BPSK, un caractère +1 produira un bit de valeur 1, etc.

4.2 Modulation BPSK

Dans le cadre de ce tp, les signaux ne seront pas modulés. Seuls les signaux en bande de base seront visualisés et observés.

Nous commencerons d'abord par l'étude de la BPSK - qui, encore une fois, est identique à une ASK-2. Pour cela, on génèrera des trains binaires qui seront pris en charge par un bloc unique appellé **PSK Mod** et qui se chargera de créer les symboles et de filtrer - et donc, de mettre ne forme le signal à l'aide d'une impulsion convenablement choisie.

Préparation 6 *Rappelez le principe de modulation de BPSK par un schéma récapitulatif.*

Manipulation 9 *En se basant sur le cours et en utilisant les blocs définis précédemment, construisez le schéma de la modulation BPSK en utilisant les blocs* ***Vector Source***, ***PSK Mod*** *et les afficheurs* ***WX GUI Scope***, ***WX GUI Fft*** *et* ***WX GUI Constellation***. *Le type de donnée des blocs Vector Source et Throttle sera le type Byte.*

a. *Dans le bloc* ***Vector Source***, *indiquez que vous cherchez à transmettre la suite binaire 010101... en renseignant le vecteur* `(0b01010101,0b01010101)` *- le bloc détectera une erreur si il n'y a qu'un élément. Vous observerez également la suite 0000111100001111... en renseignant le vecteur* `(0b00001111,0b00001111)`.

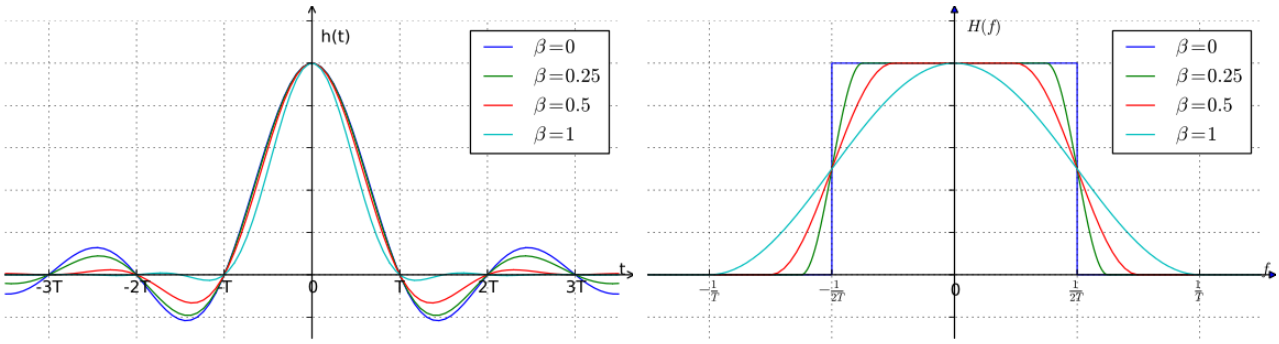
b. *Mettez un* ***Throttle*** *en indiquant une fréquence d'échantillonnage de 32 kHz. Cela limitera le nombre d'éléments binaires traités à 32000 par seconde.*

c. *Dans* ***PSK Mod***, *spécifiez les paramètres suivants : 2 points dans la constellation, pas d'encodage différentiel et 8 échantillons par symboles (le reste pouvant conserver la valeur par défaut).*

Remarque : le nombre d'échantillons par symbole correspond au nombre d'échantillons utilisés pour former l'impulsion de mise en forme. Ainsi, pour un échantillon (i.e. un symbole binaire) traité, ***il y aura 8 (par exemple) échantillons délivrés***. *Ainsi, la fréquence d'échantillonnage en sortie du bloc PSK Mod est multipliée par 8 (dans notre exemple).*

Manipulation 10 *Affichez le signal dans le domaine temporel, son contenu fréquentiel ainsi que la constellation. Expliquer les figures obtenues.*

Nous touchons quelque chose d'important ici : **l'impulsion $h(t)$ de mise en forme n'est pas une porte temporelle de durée T, mais une impulsion de Nyquist**. Pour rappel, l'impulsion de Nyquist dépend d'un paramètre de Roll-Off variant entre 0 et 1 - le paramètre Excess Bandwidth dans le bloc PSK Mod - qui modifie l'allure de $h(t)$. Une représentation de cette forme est donnée en Figure 4a tandis que la réponse fréquentielle est donnée en Figure 4b. On observe alors que ce filtre joue aussi le rôle de filtre passe-bas et limite l'étendu spectrale du signal au canal alloué.



(a) La réponse impulsionelle (b) La réponse fréquentielle

FIGURE 4 – Le filtre Root-Raised Cosine

Dans cette partie, nous allons aborder la démodulation de la BPSK.

Préparation 7 *Rappelez le principe de la démodulation BPSK.*

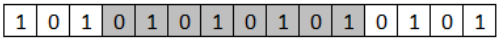
Avant de continuer, reprenons le traitement de l'information le long de la chaîne. Le bloc Vector Source délivre en boucle l'octet 0b10101010. Une fois entré dans le bloc PSK Mod, cet octet est divisé en 8 bits : 1, 0, 1, ... Chacun de ces bits est transmis indépendamment des autres (BPSK oblige) et un symbole est transmis par bit. Une fois reçu par le bloc PSK Demod, chacun de ces symbole est décodé et transmis indépendamment des autres. Ainsi, le bloc PSK Demod délivre non pas l'information initiale (i.e. 0b10101010), mais une suite d'octets n'ayant qu'un seul bit utile : 0x01, 0x00, 0x01, 0x00, ... Il faut donc ne conserver que les derniers bits de chaque octet en sortie et les rassembler dans un unique Byte. Pour cela, on utilisera le bloc Repack ou Unpacked to Packed.

Manipulation 11 *Mettez en œuvre la réception de la BPSK à l'aide du bloc* ***PSK Demod****. Vous prendrez soins à placer un bloc* ***Unpacked to Packed*** *ou* ***Repack****.*
Observez la sortie à l'aide d'un scope. Que remarquez-vous ? Le signal reconstruit est-il conforme à vos attentes ?

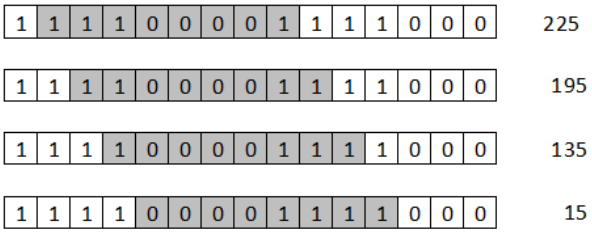
Nous pointons ici un autre problème. En sortie du bloc PSK Demod, le train binaire (ou encore une fois, chaque bit est placé à la fin d'un octet) est une succession de bits. Le bloc Unpacked To Packed choisit dans ce train binaire un bit de début de trame et concatène les bits par paquets de 8 pour former des octets. Par exemple, si la source délivre 0b10101010, alors le bloc Unpacked to Packed peut choisir de rassembler ces 8 bits

ou ceux-là ...

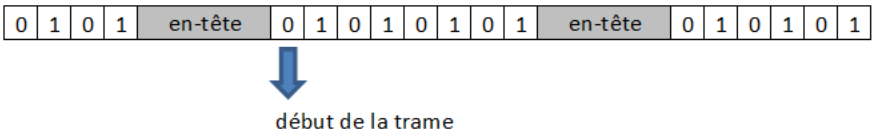
et ainsi de suite ...



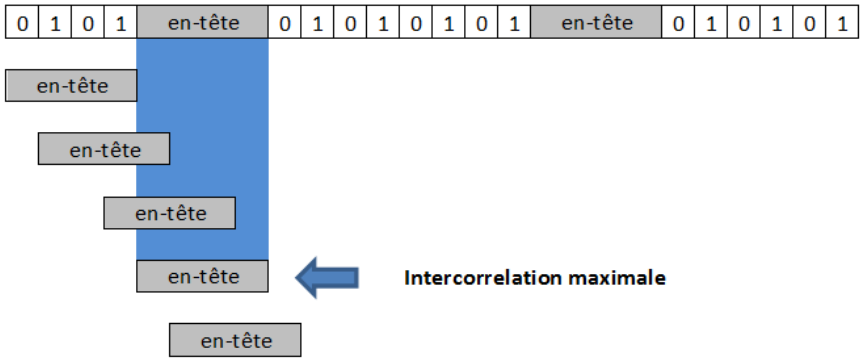
Et, dans cet exemple, le choix du premier bit ne peux mener qu'à une probabilité d'erreur de réception que de 1/2. Pour des trames plus compliquées, comme pour une source délivrant 0b00001111, l'erreur est quasi-systématique ...



On touche donc le point délicat de **la détection du début de la trame d'information**. Ce problème se résout en utilisant des successions connues de bits appelées **en-tête** régulièrement insérées dans le train binaire avant émission.



Ainsi, le récepteur peut concaténer les bons bits et restituer l'information convenablement en détectant l'en-tête. Pour cela, la méthode classique est de réaliser une intercorrelation entre le train binaire et l'en-tête connue du récepteur



Cette opération est réalisée par les blocs Packet Encoder/Decoder. Ces blocs insèrent dans le flux de donnée les en-tête connus à l'émission et à la réception qui servira de balise pour signaler le début de la transmission. Ils se placent en amont de la modulation PSK et en aval de la démodulation.

Manipulation 12 *Mettez en œuvre la transmission numérique en utilisant les blocs* ***Packet Encoder*** *et* ***Packet Decoder****. Vérifiez la bonne réception des données en affichant simultanément les échantillons émis et ceux reçus.*

4.3 Mise en œuvre de la DBPSK

La PSK mise en place dans la partie précédente a deux défauts :

— Pour l'heure, il n'y a pas de retard de transmission entre l'émetteur et le récepteur. En pratique, si le signal $x(t)$ est émis, le signal reçu est $Kx(t-\tau)$ - cf les rappels plus haut dans le document. Ainsi, le signal démodulé u_r est affecté de ce même retard : le signal démodulé est $u_r(t-\tau) = u_r(t)e^{-j2\pi f\tau}$. Ce retard, du fait du coefficient $e^{-j2\pi f\tau}$, fait tourner la constellation en réception. Or, la PSK détecte la présence d'un point de mesure au voisinage d'un point remarquable (-1, i, ...). Ainsi, si la constellation tourne, l'appareillage ne se fait plus.

— En pratique, comme nous l'avons dis dans la partie précédente, le signal en sortie du bloc PSK Mod est multiplié par la porteuse afin de former le signal à transmettre. Seulement, en réception, afin de régénérer la porteuse, une PLL est utilisée. Celle-ci - et c'est la théorie qui le montre - peut s'accrocher soit sur la phase de la porteuse, soit sur la porteuse déphasée de π. Ainsi, en réception, le symbole reçu est multiplié par 1 ou par -1. On reçoit soit le signal émis, soit son complémentaire.

Afin d'éviter ces problèmes, le signal à émettre α_k est codé de manière différentielle pour former le signal γ_k suivant la loi suivante

$$\gamma_k = \begin{cases} 0 \text{ si } \alpha_k = \alpha_{k-1} \\ 1 \text{ si } \alpha_k \neq \alpha_{k-1} \end{cases}$$

C'est ce signal différentiel qui est utilisé pour former les caractères complexes β_k - définis dans la partie précédente - en vue de former le signal modulant.

Cette opération produit la modulation DPSK comme Differential PSK.

Modulation DPSK

Dans cette partie nous allons effectuer la modulation DPSK à deux états d'un signal cosinus.

Pour rappel, nous avons vu dans le TP0 que la BPSK n'était rien d'autre qu'une ASK-2 s'obtenant facilement en multipliant le signal modulant $u(t)$ avec la porteuse.

Préparation 8 *Donnez un exemple de réalisation possible du signal modulé de la DBPSK à l'aide d'une porte logique.*

Manipulation 13 *En se basant sur le cour et en utilisant les blocs définis précédemment, construisez le schéma de la modulation DBPSK en utilisant un Vector Source et le blocs DPSK Mod.*

Manipulation 14 *Observez le spectre du signal modulé et sa constellation. Expliquez.*

Manipulation 15 *Quel est le facteur qui contrôle la largeur du spectre ?*

Manipulation 16 *Rajoutez l'affichage Waterfall à la sortie de PSK Mod. Expliquez.*

Manipulation 17 *Rajouter du bruit au signal modulé en faisant varier son amplitude. Affichez les figures pertinentes avec un niveau de bruit élevé. Que remarquez-vous ? Conclure.*

Démodulation DPSK

Dans cette partie nous allons aborder la démodulation DPSK à deux états, pour cela il est important que la fréquence de la porteuse et sa phase soient connue par le récepteur.

Préparation 9 *Donnez un schéma de réalisation possible de la démodulation DBPSK.*

Manipulation 18 *De même que dans la partie précédente, réalisez sur le même flowgraphe l'émission et la réception de symboles simple - à l'aide de Vector Source, par exemple.*

Qualité de la transmission - Diagramme de l'œil

A présent que vous disposez d'un schéma de modulation / démodulation DBPSK complet, vous allez mettre en œuvre une transmission de données plus complexes. Pour cela, vous utiliserez un bloc **Signal Source**. Vous pourrez, par exemple, émettre un signal sinusoïdal dont la fréquence est contrôlée par un GUI Slider .

Manipulation 19 *Modifiez le schéma afin de transmettre un signal sinusoïdal. Observez le signal émis et reçu ainsi que leur spectres. Assurez-vous de la bonne réception du signal.*

A présent, il va s'agir de mettre en évidence la robustesse de cette transmission face à deux détériorations du signal :

1. d'une part, le bruit ajouté au signal en traversant le milieu de propagation,
2. d'autre part, la bande passante du canal qui peut introduire de l'interférence entre symboles en cas de mauvais réglage.

Nous allons commencer par étudier le premier facteur. Pour cela, vous utiliserez le bloc **Chanel Model** qui permet de préciser le niveau d'un bruit ajouté (utilisez également un GUI Slider). D'autre part, vous prendrez soin à donner la valeur 1 (et non 1+j) au paramètre Taps.

Manipulation 20 *Modifiez le schéma afin d'ajouter du bruit au signal transmis. Observez la constellation et le signal modulé. Observez le diagramme de l'œil de la transmission. Pour cela,*

1. *vous observerez le signal modulé,*
2. *vous utiliserez un bloc Signal Source à la fréquence la rapidité des symboles,*
3. *vous synchroniserez l'oscilloscope sur le signal délivré par le bloc Signal Source,*
4. *vous effacerez tous les signaux inutiles en choisissant 'None' dans l'option Marker,*
5. *vous mettrez la persistance au maximum,*
6. *vous alternerez la synchronisation sur pente positive et négative.*

Observez la détérioration de l'ouverture de l'œil avec le niveau de bruit.

La présence de bruit ajouté par le canal augmente la probabilité de se tromper de caractère au moment de la prise de décision. En effet, pour la BPSK, les deux caractères sont -1 (pour le 0 logique) et le +1 (pour le 1 logique). En présence de bruit, un caractère valant +1 peut se rapprocher dangereusement du seuil de décision (classiquement 0 dans le cas équiprobable, une autre valeur peut être choisie si un caractère est plus probable qu'un autre afin d'améliorer la probabilité d'erreur) et être considéré comme un -1, causant une erreur. Ainsi, la conséquence de la fermeture de l'œil, c'est l'erreur de choix au moment de la décision.

Seulement, le canal peut amener une autre source d'erreur : l'interférence entre symboles. La règle de Nyquist dit : **on ne peut transmettre un information de rapidité 1/T dans un canal de bande passante inférieure à 1/2T sans avoir d'interférence entre symboles**. Seulement, vous avez vu que les bloc DPSK Mod/Demod contiennent un filtre de Root-Nyquist qui satisfait cette condition. Ainsi, jusqu'à présent, vous étiez protégé de l'interférence entre symbole. Pour le mettre en évidence, vous devez alors filtrer le signal modulé à l'aide d'un filtre passe-bas *de fréquence de coupure inférieure à 1/2T*.

Manipulation 21 *Mettez en œuvre un filtre passe-bas et un Slider pour mettre en évidence l'interférence entre symboles. Affichez le diagramme de l'œil ainsi obtenu. Observez l'effet du choix de la bande passante du canal sur la capacité à décider du symbole.*
Remarque : En rajoutant un filtre dans le milieu de transmission, il semblerait que le bloc Packet Decoder ne réussisse à retrouver le header. Cela sera peu important puisque les signal reçu nous importe peu ici.

L'interférence entre symbole rend presque indétectable les symboles en réception. En effet, chaque symbole "bave" sur son voisin.

Modulation DQPSK

Manipulation 22 *Changez le type de modulation en DQPSK. Visualisez les formes d'onde.*

4.4 Modulation QAM

Cette partie porte sur la modulation d'amplitude en quadrature QAM et se présente comme un complément aux parties précédentes.

Le format de modulation QAM est étudié très sérieusement dans le domaine des télécommunications par satellite ou par fibre optique afin de répondre aux besoins exprimés par une augmentation de plus en plus importante du débit. En effet, cette méthode permet d'augmenter singulièrement la quantité d'information transportée par chaque symbole - et cela au risque d'erreurs plus probable.

En vous basant sur les blocs définis précédemment et sur les blocs QAM Mod/Demod, vous allez concevoir un schéma de modulation et démodulation QAM pour un signal sinusoïdal.

Préparation 10 *Expliquer le principe de la QAM.*

Manipulation 23 *Mettez en œuvre la modulation MAQ pour une constellation composée de 16 points. Vous utiliserez d'abord un signal Random Source. Observez le spectre du signal modulant, la forme d'onde du signal et la constellation - on utilisera pour cela l'élément Scope, l'instrument Constellation ne fonctionne que pour les PSK.*

Manipulation 24 *Mettre en place la démodulation. Observez la bonne restitution du signal. Pour cela, utilisez les blocs Packet Encoder/Decoder.*

Manipulation 25 *Mettre en place un bloc Channel Model, ajouter du bruit sur le milieu de transmission. Observez le résultat. Conclure sur la sensibilité de ces systèmes aux bruit.*