PR214 - Rapport de projet

Prototypage d'un système de communications numériques complet à base de modules SdR

Auteurs :

Maël Laviec
Léa Volpin

Promotion :
ENSEIRB-MATMECA 2022

Référents :

Bertrand Le Gal
Guillaume Ferré

Mai 2021

Table des matières

1 Introduction

Ce projet fait suite à un projet réalisé au semestre 9 par la filière Systèmes Embarqués (SE 2020).

1.1 Objectifs

L'objectif de ce projet est de développer une chaîne de communications numériques complète pour assurer la transmission d'images ou d'un flux vidéo entre EirBalloon2 et la station radio-amateur de l'ENSEIRB-MATMECA. Le projet pourra également être utilisé comme illustration des projets réalisés par les élèves lors des portes ouvertes de l'école.

FIGURE 1 – *Module SDR HackRF One utilisé*

Dans le cadre de notre projet, l'étage analogique de la chaîne de communications numériques sera réalisé par des modules SDR (Software Defined Radio). Ils ont comme avantage d'être des systèmes RF (Radio Fréquences) configurables. Leur flexibilité (utilisation en mode émetteur Tx ou récepteur Rx) permet notamment de s'adapter à l'architecture de la chaîne de communications retenue.

1.2 Paramètres de la chaîne de communications

L'objectif du projet étant une implantation sur EirBalloon2, plusieurs choix de paramètres se sont imposés. En effet, nous pouvons considérer que le chemin des ondes électromagnétiques (EM) entre l'émetteur (le ballon) et le récepteur (la station radio) est un chemin direct, c'est-à-dire sans multitrajets de l'onde EM. Ainsi, notre premier choix d'architecture a été effectué : nos communications seront sur mono-porteuse. La modulation numérique a été fixée comme de la modulation QPSK (Quadrature Phase Shift Keying). Nous avons alors $nb = 2\,bits/symbole$. Par ailleurs, la fréquence de la porteuse a été définie comme étant $f_p = 1.2\,GHz$ sur une puissance d'émission de $P_e = 1\,W$. Afin de satisfaire des flux vidéos ou des transmissions d'images, le débit binaire est de $D_b = 1\,Mbits/s$. Nous avons alors un débit symbole de $D_s = \frac{D_b}{nb} = 500\,kSymboles/s$ ce qui implique un temps symbole de $T_s = \frac{1}{D_s} = 2\,\mu s$. De plus, les CAN et les CNA ont un temps d'échantillonnage fixe : $T_e = 0,1\,\mu s$. Nous avons alors le facteur de sur-échantillonnage suivant : $F_{se} = \frac{T_s}{T_e} = 20$

En résumé, nous avons les paramètres suivants :
- type de la chaîne de communications : **mono-porteuse** ;
- modulation numérique : **QPSK** ;
- fréquence de la porteuse : $f_p = 1,2\,GHz$;
- puissance d'émission : $P_e = 30\,dBm$;
- débit binaire : $D_b = 1\,Mbits/s \Rightarrow D_s = 500\,kSymboles/s \Rightarrow T_s = 2\,\mu s$.
- facteur de sur-échantillonnage : $F_{se} = 20 \Rightarrow f_e = 10\,MHz$;

2 Approche théorique

Dans cette partie, nous allons décrire le modèle théorique de notre chaîne de communications. Nous allons également prendre en compte les complications dues aux transmissions réelles (bruit, limite de l'électronique etc.). Dans un premier temps, nous décrivons chaque bloc qui compose notre émetteur et récepteur. Par la suite, nous allons présenter les algorithmes que nous avons retenus pour pallier à ces complications. Enfin, nous allons apporter un regard critique sur les solutions apportées.

2.1 Modélisation de la chaîne de communications réelle

Comme énoncé dans la partie §1.3, notre chaîne de communications est sur mono-porteuse. Ainsi, son architecture est semblable à :

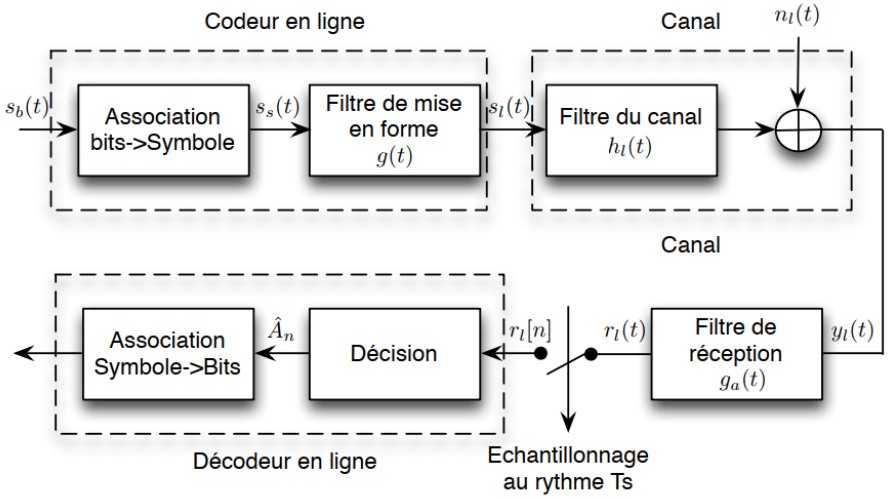

FIGURE 2 – *Chaîne de communications numériques mono-porteuse*

Il s'agit ici de la chaîne basique. La partie émission (Tx) reste relativement identique dans le cadre de notre projet. Cependant, la partie réception (Rx) présente de nouveaux blocs de traitement numérique afin de minimiser les effets indésirables d'une transmission réelle. On notera donc l'apparition d'un bloc TNDS (Traitement Numérique Du Signal) qui permet, entre autre, de réaliser une pré-synchronisation fréquentielle et temporelle (voir la section §2.2 sur les algorithmes retenus).

2.1.1 Émetteur

Dans cette partie, les différents principes physiques et équations qui régissent notre émetteur sont présentés. A l'entrée de notre chaîne nous avons un flux binaire (dans notre cas, le flux binaire représente celui de l'image *background.jpeg*) qui peut être traduit par l'équation ci-dessous :

$$s_b(t) = \sum_{k=1}^{N_b} b_k \delta(t - kT_b) \tag{1}$$

Où $T_b = 1,0.10^{-6}\, s$ est la période binaire. Nous avons volontairement posé comme indice initial $k = 1$ au lieu de $k = 0$ afin de faciliter la modélisation sur MATLAB.

Puisque nous avons choisi une modulation QPSK, nous avons la répartition des combinaisons binaires en symboles suivante :

combinaison binaire b_k et b_{k+1}	symbole associé S_k
00	$e^{j\frac{\pi}{4}}$
10	$e^{j\frac{3\pi}{4}}$
11	$e^{j\frac{5\pi}{4}}$
01	$e^{j\frac{7\pi}{4}}$

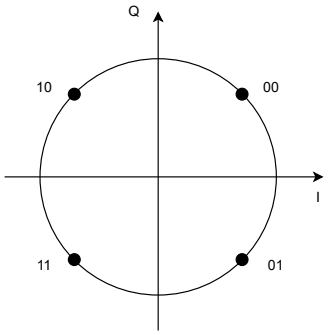

FIGURE 3 – *Répartition des symboles en fonction des combinaisons binaires avec leur représentation dans le plan complexe (voies I et Q)*

Ainsi, nous déduisons l'expression du flux de symbole à la sortie de modulateur numérique :

$$s_s(t) = \sum_{k=1}^{N_b/2} S_k \delta(t - kT_s) \tag{2}$$

Et celle à la sortie du filtre de mise en forme :

$$s_l(t) = (g * s_s)(t) = \sum_{k=1}^{N_b.Fse/2} S_k g_x(t - kT_s) \tag{3}$$

Avec $g_x(t)$ le filtre de mise en forme choisi. Dans notre cas, nous avons utilisé le filtre porte puis le filtre en racine de cosinus surélevé (rcos) avec un roll-off $\alpha = 0,8$. Ils sont définis par :

$$g_{porte}(t) = \begin{cases} 1 \text{ si } 0 \leq t < T_s \\ 0 \text{ sinon} \end{cases} \quad \text{et} \quad g_{rcos}(t) = \begin{cases} \dfrac{4\alpha}{\pi\sqrt{T_s}} \dfrac{cos\left(\pi\frac{(1+\alpha)t}{T_s}\right) + \frac{sin\left(\pi\frac{(1-\alpha)t}{T_s}\right)}{\frac{4\alpha t}{T_s}}}{1-(\frac{4\alpha t}{T_s})^2} \text{ si } -4T_s \leq t < 4T_s \\ 0 \text{ sinon} \end{cases}$$

Le diagramme de l'oeil des deux signaux $s_l(t)$ pour les différents filtres de mise en forme sont représentés ci-dessous :

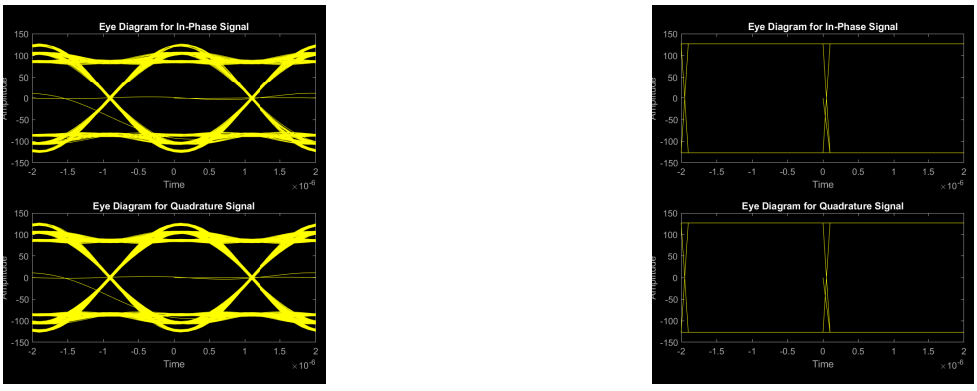

FIGURE 4 – *Diagramme de l'oeil pour un filtre de mise en forme en racine de cosinus surélevé avec $\alpha = 0,8$ (à gauche) et un filtre porte (à droite) du signal $s_l(t)$*

Par la suite, le signal $s_l(t)$ est monté à la fréquence porteuse avec l'étage analogique du module SDR (non représenté sur la figure 2). L'expression à la sortie de l'émetteur est alors semblable (pour filtre de mise en forme en rcos) à :

$$s(t) = \sum_{n=1}^{N_b.Fse/2+4F_{se}} \sqrt{\frac{P_e}{P_{s_l}}}.S_k.g_x(t-(k+4)T_s).exp[j(2\pi(f_p + \Delta f_{Tx})t)] \tag{4}$$

Où P_{s_l} est la puissance du signal $s_l(t)$, Δf_{Tx} est le décalage de fréquence de entre l'oscillateur local du module SDR Tx et la fréquence théorique de la porteuse f_p et $k = \left\lfloor \frac{n}{F_{se}} \right\rfloor$.

La formulation théorique du signal en sortie nous sera utile pour la compréhension de certains phénomènes physiques (décalage fréquentiel et temporel).

2.1.2 Canal de propagation

Nous avons précédemment décrit le signal en sortie de notre émetteur. Ce signal se propage dans l'air sur une certaine distance. On estime, avec le vol de EirBalloon1, une distance maximale entre Tx/Rx de $R_{max} = 300\,km$. Une telle distance a bien entendu des répercutions sur la qualité du signal reçu. En effet, il est majoritairement soumis à deux phénomènes :

— le bruit thermique de l'atmosphère terrestre $n(t)$; On pose ici que le bruit $n(t)$ est un bruit blanc gaussien additif tel que :

$$n(t) \sim \mathcal{N}_{\mathcal{C}}(0,\, \sigma_{nl}^2)$$

— les pertes en espace libre $L_s(R)$:

$$L_s(R) = \left(\frac{4\pi R}{\lambda}\right)^2 \quad \text{où } \lambda = \frac{c}{f_p} = 0,25\,m$$

Ainsi, l'expression à l'entrée du récepteur (pour filtre de mise en forme en rcos) est :

$$y(t) = \sum_{n=1}^{N_b.Fse/2+4F_{se}} \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}.S_k.g_x[t-(k+4)T_s].exp[j(2\pi(f_p+\Delta f_{Tx})t)] + n(t) \tag{5}$$

Nous nous sommes également intéressés à l'évolution du rapport signal sur bruit : SNR (Signal Noise Ratio) avec les paramètres de notre chaîne de communications. On décrit ainsi l'évolution de notre SNR en fonction de la distance Tx/Rx telle que :

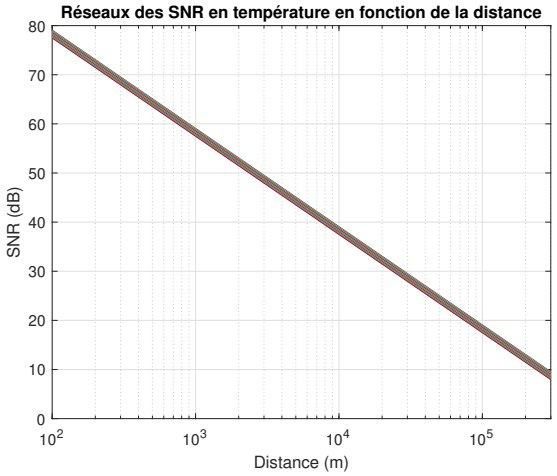

FIGURE 5 – *Rapport signal sur bruit avec $G_{Tx} = 8\,dBi$ et $G_{Rx} = 0\,dBi$ pour $T_{atmos} \in [-50; 50]°C$*

On constate alors que le rapport SNR le plus faible (donc pour la température la plus forte et la distance la plus longue) vaut :

$$SNR = \frac{E_b}{N_0} = 7,94\,dB$$

Pour $R = 300\,km$ et $T_{atmos} = 50\,°C$.

À partir de ce résultat, nous imposons un fonctionnement total (c'est-à-dire une probabilité d'erreur très faible) de notre chaîne de communications lorsque $SNR > 5\,dB$.

Malheureusement, une autre complication voit le jour : le temps de propagation du signal dans l'atmosphère. Bien que la distance maximale soit "courte" pour la célérité des ondes EM, elle est suffisamment importante pour engendrer un dysfonctionnement du récepteur (d'autant plus que nous ne savons pas quand nous allons recevoir une communication). En supposant qu'il y ait une transmission continue de son décollage jusqu'à son atterrissage, le retard s'exprime comme :

$$\tau = \frac{R}{c} = 1\,\mu s \quad \text{pour } R_{max} = 300\,km$$

Nous aboutissons alors à l'expression finale équivalente du signal à l'entrée de notre récepteur (pour un filtre de mise en forme en racine de cosinus surélevé) :

$$y'(t) = y(t-\tau) = \sum_{n=1}^{N_b.Fse/2+4F_{se}} \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}.S_k.g_x[t-(k+4)T_s-\tau].exp[j(2\pi(f_p+\Delta f_{Tx})(t-\tau))] + n(t) \quad (6)$$

Nous distinguons alors tout l'enjeu de détecter la présence du signal et à quel moment il commence. Par ailleurs, nous pouvons prévoir une désynchronisation fréquentielle dû au terme Δf_{Tx}. Par conséquent, notre objectif est de venir au mieux estimer les paramètres τ et Δf_{Tx} (qui sera également constitué de Δf_{Rx}).

2.1.3 Récepteur

Le retour en Bande de Base (BdB) de notre signal $y'(t)$ s'effectue grâce à un module SDR configuré en récepteur. Malheureusement, le récepteur, notamment l'oscillateur local, n'est pas parfait et un nouveau décalage est constaté : Δf_{Rx}. Ainsi, le décalage fréquentiel résultant entre l'émetteur et le récepteur est :

$$\Delta f = \Delta f_{Tx} + \Delta f_{Rx}$$

Dans notre bloc TNDS, nous devrons donc venir au mieux estimer la valeur de ce décalage fréquentiel Δf résultant des défauts du Tx et Rx. Deux méthodes seront présentées dans la partie §2.2.2.

Ensuite, le signal est filtré par son filtre adapté afin de maximiser le SNR. Nous avons alors :

$$v_x(t) = (g_x * g_{x,a})(t)$$

Où $g_{x,a}(t)$ est le filtre adapté du filtre en rcos ou porte. Ainsi, dans le cadre d'un filtre de mise en forme rcos, l'expression à la sortie du filtre adapté, donc à l'entrée du sous-échantillonnage, est :

$$r_l(t) = \sum_{n=1}^{N_b.Fse/2+8F_{se}} \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}.S_k.v_x[t-(k+8)T_s-\tau].exp[j(2\pi\Delta f(t-\tau))] + n'(t) \quad (7)$$

où $n'(t) = (n * g_{x,a})(t)$ est le bruit convolué au filtre de mise en forme adapté.

Nous avons choisi ici des filtres de mise en forme respectant le critère de Nyquist de telle manière que le k-ième échantillon (après sous-échantillonnage d'un facteur F_{se}) soit le k-ième symbole :

$$r_l[k] = v[0]S_k + n'[k]$$

Or si nous effectuons un sous-échantillonnage d'un facteur F_{se} sur $r_l(t)$, nous obtenons le signal suivant :

$$r_l[k] = \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}.v_x[-\tau].exp(j(2\pi\Delta f\tau))S_k + \sum_{i=1,i\neq k}^{N_b/2} \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}.S_i.v_x[k-iT_s-\tau].exp[j(2\pi\Delta f(k-\tau))] + n'[k] \quad (8)$$

Afin de simplifier l'expression ci-dessus, nous posons $\beta(R) = \frac{1}{L_s}\sqrt{\frac{P_e}{P_{s_l}}}$, le coefficient d'atténuation du signal après propagation et θ un déphasage constant de notre signal. Nous obtenons finalement :

$$r_l[k] = \beta(R).v_x[-\tau].exp(j(2\pi\Delta f\tau + \theta))S_k + \sum_{i=1, i\neq k}^{N_b/2} \beta(R).S_i.v_x[k - iT_s - \tau].exp[j(2\pi\Delta f(k-\tau) + \theta)] + n'[k] \quad (9)$$

Bien que complexe, l'équation nous illustre principalement l'apparition d'Interférences Entre Symboles (IES), notamment avec l'apparition du bruit et de la somme des symboles adjacents au k-ième symbole. Par ailleurs, nous avons une diminution de la mise en valeur du k-ième symbole à cause du terme d'atténuation $\beta(R)$, d'un déphasage fréquentiel Δf et d'un déphasage constant θ. Ces termes vont venir perturber la qualité de la constellation après sous-échantillonnage et ainsi augmenter la probabilité d'erreur binaire comme illustré par les figures ci-dessous :

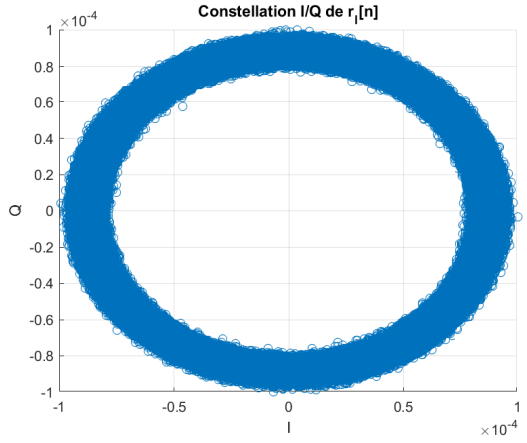

FIGURE 6 – *Constellation I/Q du signal* $r_l[k]$ *avec* $\Delta f = 94,419\,kHz$ *et* $\beta(R) = 1,0.10^{-4}$

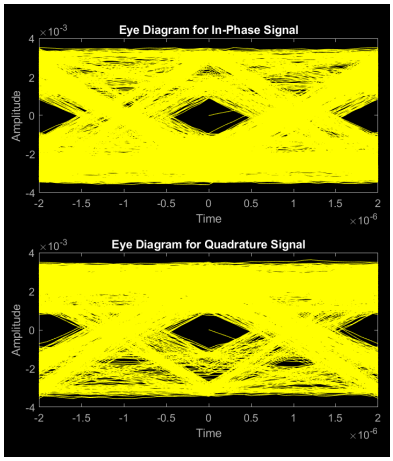

FIGURE 7 – *Diagramme de l'oeil du signal* $r_l[k]$ *avec* $\Delta f = 94,419\,kHz$ *et* $\beta(R) = 1,0.10^{-4}$

L'équation (9), la figure 6 et la comparaison de figure 4 avec la 7 attestent de la nécessité de faire une correction de notre signal reçu afin d'en ressortir un signal exploitable pour la suite de la chaîne de communications. En effet, nous souhaitons transmettre des images au format *.jpeg*, celui-ci n'est plus exploitable si un seul de ses bits est erroné. Nous devons donc impérativement implémenter de nouvelles fonctionnalités afin d'assurer la transmission sans embûches.

2.2 Algorithmes retenus

Dans cette partie, nous allons présenter les solutions que nous avons retenues pour notre chaîne de communications. Nous avons raisonné sur deux axes : une synchronisation temporelle et une synchronisation fréquentielle.

2.2.1 Synchronisation fréquentielle

Le but de la synchronisation fréquentielle est de venir estimer au mieux Δf mais également de supprimer le déphasage constant θ. Nous avons trouvé deux solutions :

— par recherche de la fréquence du maximum de $r_l^4[k]$ puis soustraction de la fréquence trouvée ;

— par codage différentiel des symboles complexes.

La première solution est une technique astucieuse. En effet, la QPSK peut être vue comme une variable aléatoire de la forme :

$$e^{j\phi(t)\frac{\pi}{4}} \quad \text{où} \quad \phi(t) \in 1, 3, 5, 7$$

Ainsi, si nous élevons cette expression à la puissance 4, nous avons :

$$e^{j4\phi(t)\frac{\pi}{4}} = e^{j\phi(t)\pi} = -1 \ \ \forall \phi(t)$$

Par conséquent, si nous avons une fréquence résiduelle dans notre signal $r_l(t)$ (avant sous-échantillonnage), on a le résultat ci-dessous :

$$e^{j(4\phi(t)\frac{\pi}{4}+2\pi 4\Delta f t)} = e^{j\phi(t)\pi}e^{j2\pi 4\Delta f t} = -e^{j2\pi 4\Delta f t}$$

D'où une mise en évidence prononcée de $f = 4\Delta f$:

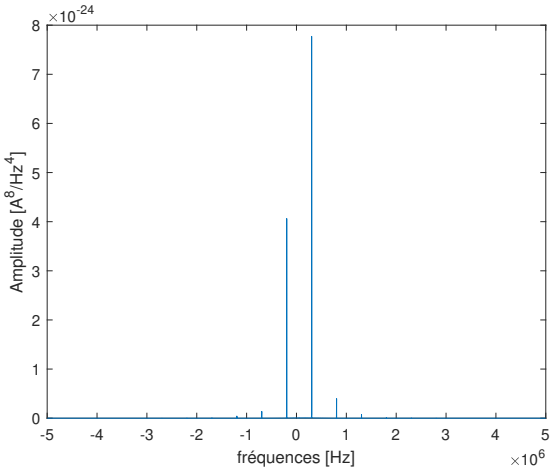

FIGURE 8 – *Transformée de Fourier discrète du signal $r_l(t)$ avec $N_{FFT} = 32768\,bins$*

Dans l'exemple de la figure 8, nous avons décalage fréquentiel de $\Delta f = 76,081\,kHz$. Or la recherche de la fréquence associée au maximum de la FFT nous donne $4\Delta f_{FFT} = 304,260\,kHz \Leftrightarrow \Delta f_{FFT} = 76,065\,kHz$.
Nous remarquons alors que la méthode nous donne une bonne approximation de la fréquence résiduelle. Toutefois, il ne s'agit pas de la valeur exacte de Δf. On explique cette différence par l'imprécision due à la FFT. En effet, les fréquences sont balayées par pas de $\frac{f_e}{N_{FFT}}$. Ainsi, la fréquence de décalage estimée peut être, dans le pire des cas, égale à :

$$\Delta f_{FFT} = \Delta f \pm \frac{f_e}{2N_{FFT}}$$

On se pose alors la question de l'efficacité de notre technique. En effet, nous substituons un décalage fréquentiel par un autre décalage fréquentiel. Si nous faisons une application numérique, on a :

$$\Delta f - \Delta f_{FFT} = 16,87\,Hz$$

Ce qui est très inférieur au décalage originel Δf. Ainsi, nous pouvons qualifier notre technique de méthode de pré-synchronisation fréquentielle car l'asservissement en fréquence n'est pas total. Par exemple, pour un transfert de $N_b = 1\,Mbits$ (à une vitesse de $1\,Mbits/s$), la constellation du récepteur à le temps de faire $2\pi(\Delta f - \Delta f_{FFT}) \cdot 1 \approx 106$ tours du cercle trigonométrique, le décodage des symboles n'est alors pas fiable. Par conséquent, cette méthode peut être considérée comme de la synchronisation fréquentielle qu'à condition que le nombre de bits à transmettre soit faible, c'est-à-dire pour un temps de transfert très faible. Nous imposons comme condition que la constellation ne doit pas être déphasée de plus de 10°, cela implique, dans le pire des cas, que :

$$N_b \leq 2\frac{10°}{180°}\frac{f_s}{f_e}N_{FFT} \approx 182\,bits \quad \text{avec } N_{FFT} = 32768\,bins$$

Un résultat qui démontre le manque d'efficacité de cet algorithme. Ainsi, nous avons envisagé une autre solution : le codage différentiel.

Le principe du codage différentiel est de venir coder le symbole de rang n en fonction des symboles précédents. Ce codage a la caractéristique de corriger (jusqu'à une certaine limite) le décalage fréquentiel. Il est alors régi par une équation récursive. Par conséquent, la suite des symboles différentiels doit être initialisée. Ainsi, en connaissant le symbole initial, nous pouvons comparer le premier symbole reçu avec le premier symbole théorique. Cette comparaison nous apporte alors une estimation de θ. Pour notre étude, la relation de récurrence a été définie telle que :

$$S_d[n+1] = S[n]S_d[n] \tag{10}$$

Avec $S_d[1] = e^{j\frac{\pi}{4}}$.

Toutefois, cette méthode a un prix. En effet, on s'aperçoit que la constellation QPSK n'est plus. En conséquence de la nouvelle méthode de codage des symboles, la constellation est semblable à une constellation 8-PSK :

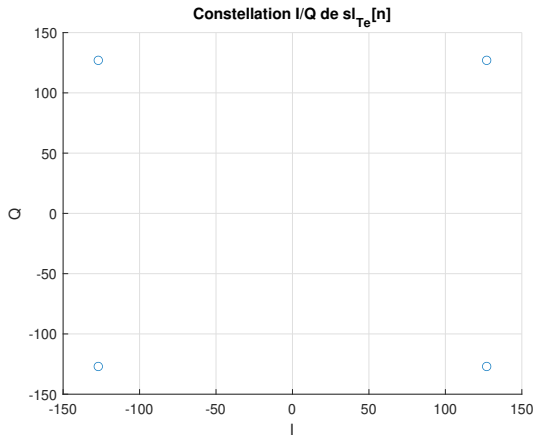

FIGURE 9 – *Constellation I/Q du signal $s(t)$ QPSK sans codage différentiel*

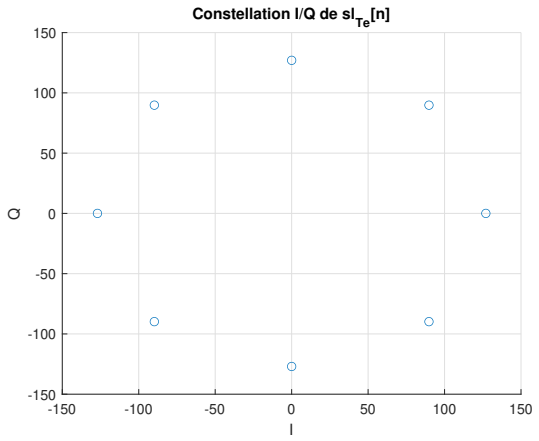

FIGURE 10 – *Constellation I/Q du signal $s(t)$ QPSK avec codage différentiel*

On en déduit trois points négatifs :

— le signal $s(t)$ devient plus sensible au bruit avec le codage différentiel ce qui en gendre des répercutions sur le taux d'erreur binaire. On observe une augmentation des probabilités d'erreurs pour un même rapport SNR par rapport à un codage QPSK classique ;

— les symboles n'étant plus indépendants, le spectre du signal $s(t)$ est déformé. Il est alors possible que la bande passante du signal soit modifiée ;

— la correction fréquentielle n'est efficace que si $2\pi\Delta f T_s \leq \frac{\pi}{4} \Leftrightarrow \Delta f \leq 62,500\,kHz$.

Malgré ces 3 défauts, la correction fréquentielle par codage différentiel semble idéale pour notre projet. En effet, si nous combinons la première méthode avec la seconde méthode, nous pouvons effectuer une bonne correction. Elle n'est toutefois pas robuste aux déviations électroniques pour des grands paquets de données.

2.3 Synchronisation temporelle

Le signal ayant un temps de propagation non nul, le signal reçu est décalé temporellement. L'enjeu est alors d'estimer le début de la trame de transmission. Tant que le retard est inférieur à T_s, l'estimation peut être faite à l'aide du critère de Garner. Toutefois, si ce retard τ est supérieur à T_s, l'estimation est plus complexe à réaliser. Nous avons envisagé une trame de transmission munie d'un préambule. Le préambule a été pensé de telle sorte qu'il est impossible d'avoir une combinaison de symboles ressemblante au préambule. Cet unicité est marquée par l'apparition de symboles nuls sur plusieurs périodes T_s (il est possible, après le filtre adapté, d'avoir une valeur nulle du signal mais elle n'est possible que sur une seule période T_s). Nous avons alors les constellation QPSK et DQPSK à la sortie du récepteur suivantes :

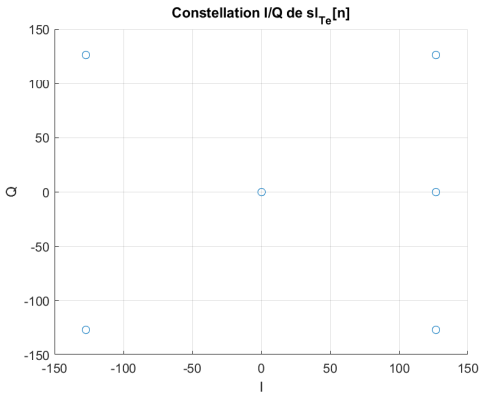

FIGURE 11 – *Constellation I/Q du signal $s(t)$ QPSK avec préambule*

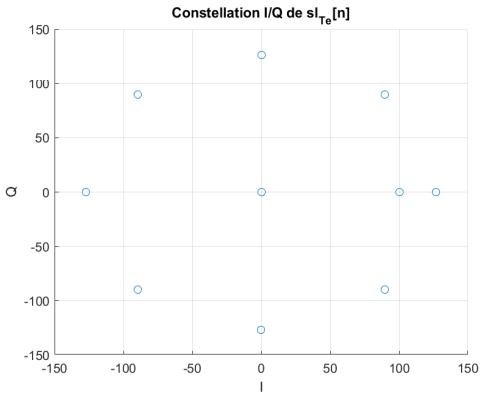

FIGURE 12 – *Constellation I/Q du signal $s(t)$ DQPSK avec préambule*

Une fois l'unicité du préambule établie, il faut détecter la présence de ce dernier. Nous avons choisi une approche par intercorrélation du signal échantillonné reçu avec le préambule théorique. Soit $s_p(t)$ le signal associé au préambule de longueur $N_p = \lfloor \frac{T_p}{T_e} \rfloor$, l'intercorrélation $\rho(u)$ est définie par :

$$\rho(u) = \frac{\sum_{n=0}^{N_p-1} y[(n+u)T_e]s_p[nT_e]}{\sqrt{E_y}\sqrt{E_{sp}}} \leq 1 \tag{11}$$

Où $E_y = \sum_{u=0}^{N_p-1} y^2[uT_e]$ est l'énergie du signal reçue et $E_{sp} = \sum_{u=0}^{N_p-1} s_p^2[uT_e]$ est l'énergie du préambule. Enfin, nous avons fixé un seuil de détection à partir duquel on considère qu'il y a présence d'un signal. La valeur a été fixé à $\gamma = 0.7$. Nous avons alors comme courbes caractéristiques les figures ci-dessous :

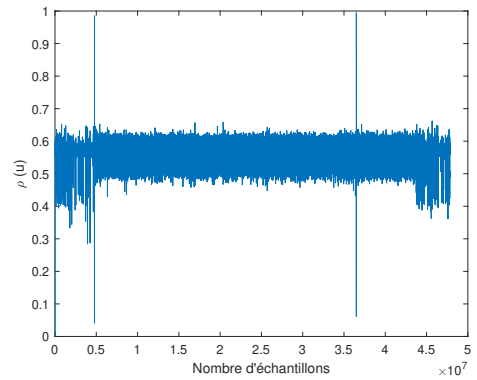

FIGURE 13 – *Intercorrélation du signal synthétique reçu avec le préambule théorique, détection de 2 préambules*

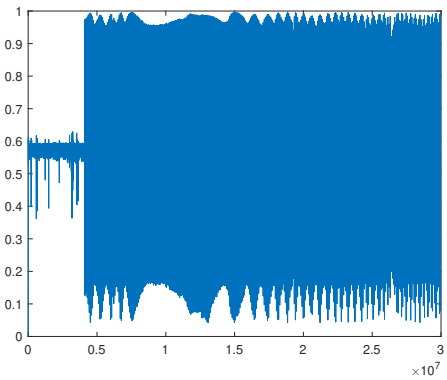

FIGURE 14 – *Intercorrélation du signal synthétique reçu avec le préambule théorique, détection de 272 préambules*

Nous pouvons constater que la détection des préambules est franche. Bien que l'intercorrélation semble être idéale, elle ne peut que fournir un indice entier du décalage temporel. Ainsi, si l'instant idéal du symbole est situé entre deux échantillons consécutifs, il ne nous n'est pas possible de déterminer la valeur intermédiaire, une solution avec une PLL est à envisager.

3 Simulations

Pour vérifier les hypothèses établies dans la théorie, nous avons effectué différentes simulations. Nous avons d'abord vérifié que nos constellations sont correctes sans bruit. Ici, nous nous concentrons sur les simulations d'un seul paquet avec les deux filtres, en mode différentiel ou non et avec du bruit dans tous les cas : rapport $SNR = 7\,dB$.

3.1 Filtre porte

3.1.1 QPSK classique

Lorsque nous faisons une modulation QPSK, à l'émetteur, les simulations sont relativement cohérentes avec la théorie. Sur la constellation nous observons les 4 symboles classiques de la QPSK et deux autres liés au préambule. En effet, le préambule n'est composé que de 0 et de 1, sa représentation se fait sur l'axe des réels. De plus, la densité spectrale de puissance (DSP) simulée suit la DSP théorique au niveau du lobe principal mais s'en éloigne de plus en plus au fur et à mesure que l'on monte en fréquence. La bande passante mesurée est $B = 1,0\,MHz$. De plus, on remarque que la DSP théorique présente des pics qui tendent vers -∞. Ils sont dûs aux zéros du sinus cardinal ($20log(0) = -\infty$). Cependant, la DSP simulée ne les possède pas. Ceci s'explique par une largeur différente entre le signal d'entrée et le filtre porte qui entraîne la suppression des zéros lors de la convolution. Finalement, nous nous apercevons d'une raie centrée en $f = 0\,Hz$. On l'explique par l'espérance non-nulle des symboles due au préambule.

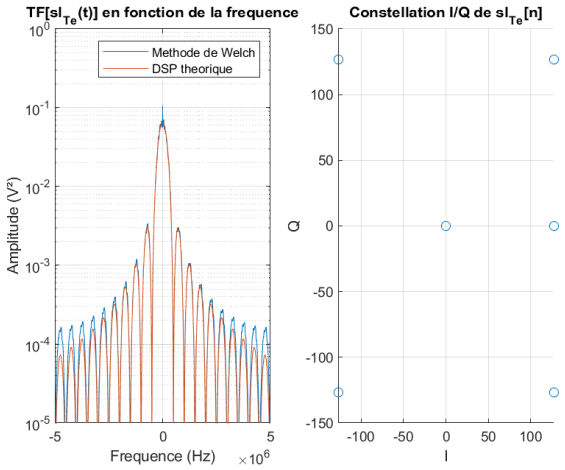

FIGURE 15 – *Transformée de Fourier discrète du signal et sa constellation à l'émetteur*

Au niveau du récepteur, on observe, comme prédit dans la théorie, que la constellation a tourné. Il y a donc une fréquence résiduelle. La moitié des bits sont alors faux : $r = 0,5$. De plus, notre bande passante a diminué : $B = 0,93\,MHz$. Il nous faut donc introduire le codage différentiel pour enlever cette fréquence résiduelle.

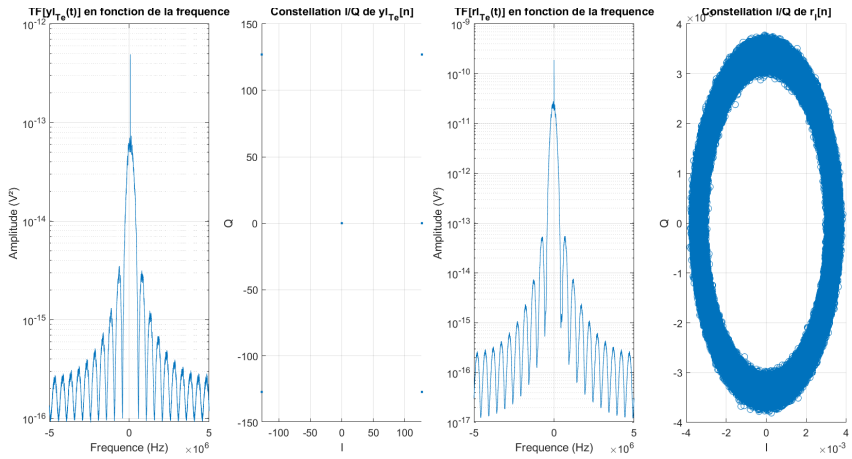

FIGURE 16 – *Transformée de Fourier discrète du signal et sa constellation au récepteur*

3.1.2 Mode différentiel

Désormais nous faisons une modulation QPSK en mode différentiel. Nous observons à l'émetteur 8 symboles semblables à ceux d'une 8-PSK et deux autres liés au préambule comme précédemment. Cette fois-ci, les symboles sont dépendants des précédents ce qui induit une déformation de la DSP. Notre bande passante est légèrement plus petite que la précédente : $B = 0,99\,MHz$.

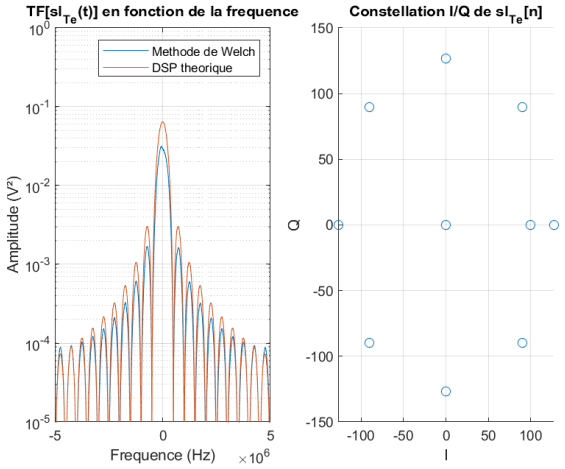

FIGURE 17 – *Transformée de Fourier discrète du signal et sa constellation à l'émetteur*

Au niveau du récepteur, cette fois-ci, la constellation n'a pas tourné. Nous avons donc réussi à enlever la fréquence résiduelle précédente. On peut cependant observer la présence du bruit sur la constellation. Après le bloc de décision, malgré le bruit, nous obtenons un taux d'erreur binaire nul : $r = 0$. Par ailleurs, notre bande passante a augmenté : $B = 1,16\,MHz$.

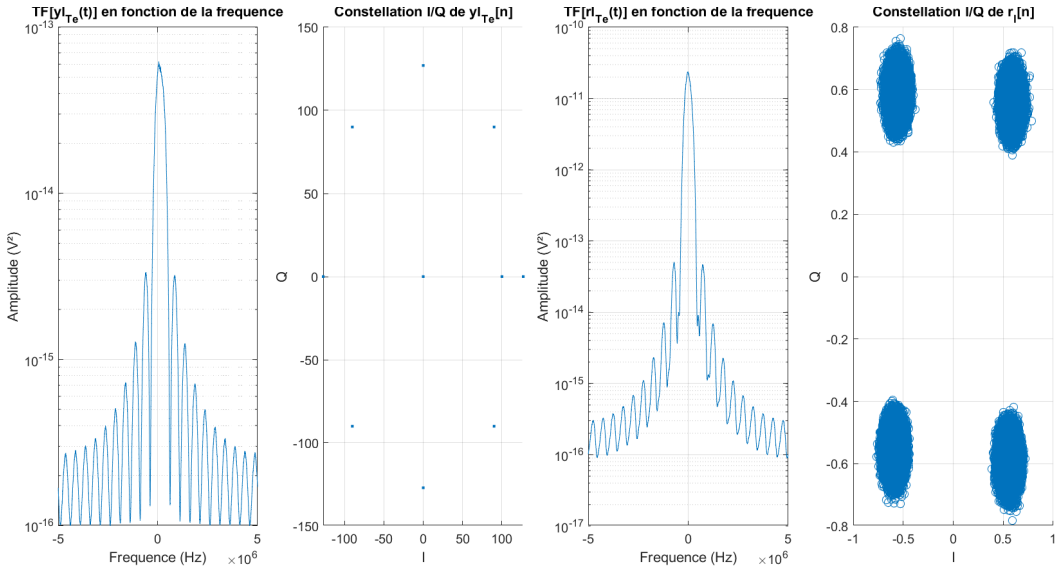

FIGURE 18 – *Transformée de Fourier discrète du signal et sa constellation au récepteur*

3.2 Filtre en racine de cosinus surélevé

3.2.1 QPSK classique

Nous effectuons une modulation QPSK classique. Au niveau de l'émetteur, sur la constellation, nous observons les symboles classiques de la QPSK mais aussi les zéros (qui ont été ajoutés lors du sur-échantillonnage) entre les symboles qui ont interagi avec les variations du filtre en rcos. On observe aussi les 2 autres symboles liés au préambule. De plus, on remarque que pour les plus basses fréquences, la DSP de la simulation suit la DSP théorique. Cependant, lorsque l'on dépasse la bande passante, la DSP théorique est complètement nulle et donc tend vers $-\infty$ en semilog alors que la DSP expérimentale ne s'annule pas mais tend vers de petites valeurs. La bande passante mesurée est $B = 0,92\,MHz$.

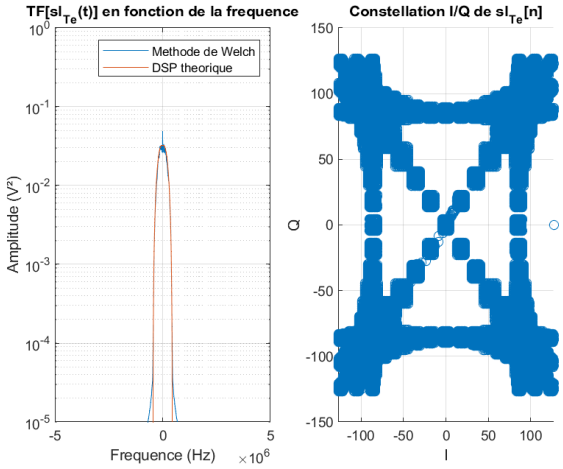

FIGURE 19 – *Transformée de Fourier discrète du signal et sa constellation à l'émetteur*

Au niveau du récepteur, comme pour le filtre porte, la constellation a tourné. Il y a donc encore une fréquence résiduelle. De plus, la moitié des bits sont faux : $r = 0,5$. Par ailleurs, notre bande passante a diminué : $B = 0,85\,MHz$. Il nous faut donc introduire le codage différentiel pour enlever cette fréquence résiduelle.

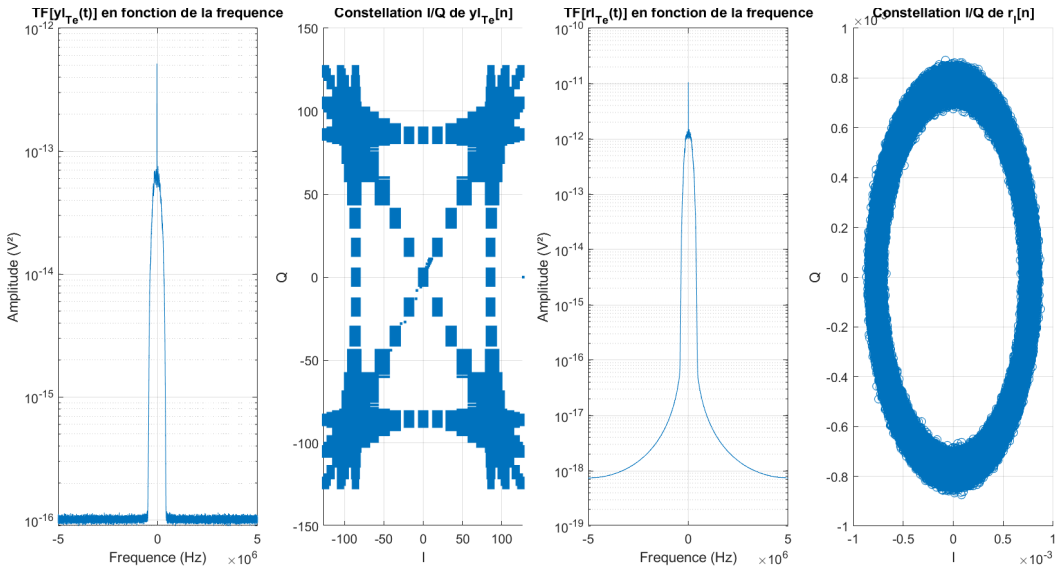

FIGURE 20 – *Transformée de Fourier discrète du signal et sa constellation au récepteur*

3.2.2 Mode différentiel

Désormais nous faisons une modulation QPSK en mode différentiel. Nous observons à l'émetteur, comme pour la simulation avec le filtre porte, les 8 symboles semblables à ceux d'une 8-PSK et les deux autres liés au préambule avec le même phénomène de modification de l'amplitude. De même, les symboles sont dépendants des précédents ce qui induit une déformation de la DSP. Notre bande passante est plus grande que la précédente : $B = 0,93\,MHz$.

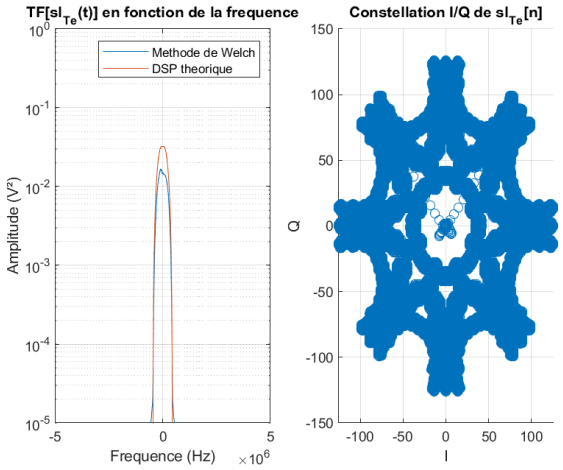

FIGURE 21 – *Transformée de Fourier discrète du signal et sa constellation à l'émetteur*

Au niveau du récepteur, comme pour la simulation avec le filtre porte, la constellation n'a pas tourné ce qui implique que la fréquence résiduelle n'est plus présente. On observe encore la présence du bruit sur la constellation mais après le bloc de décision, nous obtenons encore un taux d'erreur binaire nul : $r = 0$. Par ailleurs, notre bande passante est la même : $B = 0,93\,MHz$.

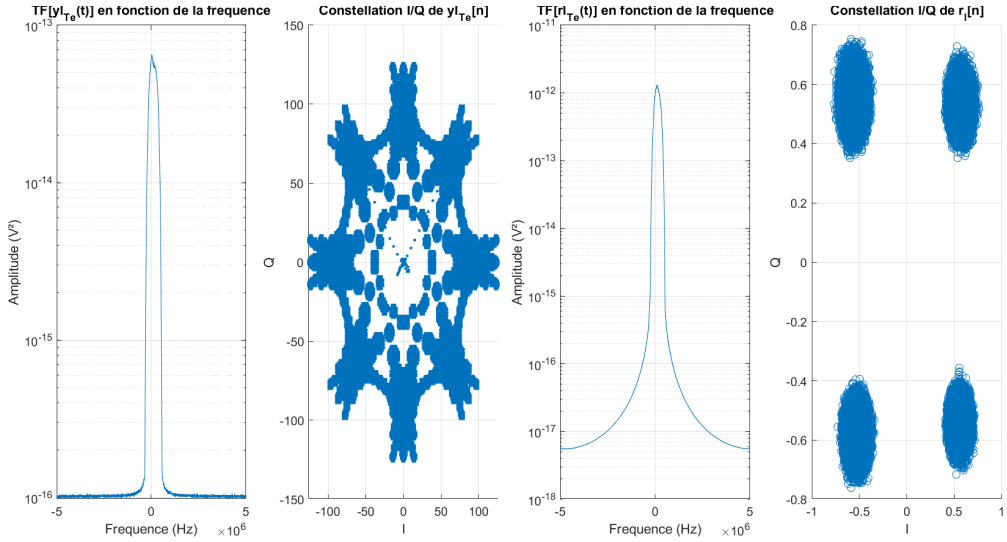

FIGURE 22 – *Transformée de Fourier discrète du signal et sa constellation au récepteur*

Après les simulations, nous comprenons l'importance du mode différentiel. En effet, il permet d'enlever au maximum la fréquence résiduelle et donc d'avoir un meilleur taux d'erreur binaire. De plus, nous favorisons un filtre en rcos car sa DSP ne présente pas de lobes secondaires. Ces derniers sont responsables de la transmission d'une partie du signal sur d'autres fréquences ce qui est énormément gênant puisque les transmissions doivent être faites sur des bandes de fréquences précises.

4 Étude de la transmission faite avec les HackRF

Une fois les simulations effectuées, nous décidons de transmettre une image grâce aux deux modules SDR dont nous disposons. Pour cela, il faut que nous "transformons" l'image en un flux de symboles modulés avec une QPSK en mode différentiel. On se sert alors de notre script Émetteur. Nous utilisons un filtre en racine de cosinus avec un roll-off $\alpha = 0,8$. Nous n'avons qu'un seul préambule et nous fixons le seuil de détection à $0,8$. Nous envoyons notre flux binaire avec le premier SDR avec le gain du VGA à $28\,dB$ et nous le recevons avec le deuxième SDR avec le gain du LNA à $24\,dB$. Ensuite, on se sert de notre script récepteur pour démodulés les symboles reçus et les remettre sous forme d'une image. Voici l'image envoyée et l'image reçue :

FIGURE 23 – *Image avant transmission*

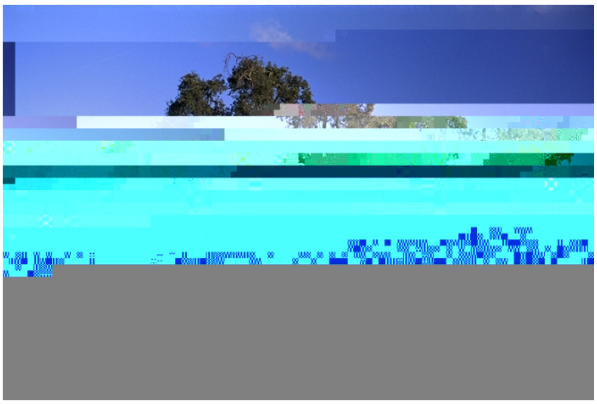

FIGURE 24 – *Image après transmission*

On remarque que seul le début de l'image est correctement transmis. Au début, la détection du préambule et la correction de phase s'effectuent. Cependant, le signal finit par se désynchroniser ce qui explique pourquoi nous n'avons pas la suite de l'image. En effet, le matériel que nous utilisons introduit un bruit de phase et les oscillateurs locaux n'oscillent pas avec une fréquence stable. Pour palier à ce problème, une solution est d'ajouter des préambules juste avant que nous perdions la synchronisation.

La constellation des symboles qui arrive au récepteur a tourné. Nous avons effectivement perdu la synchronisation. On observe bien la présence du bruit sur la constellation des symboles après passage dans le filtre adapté. Par ailleurs, notre bande passante est la même que dans les simulations : $B = 0,93\,MHz$.

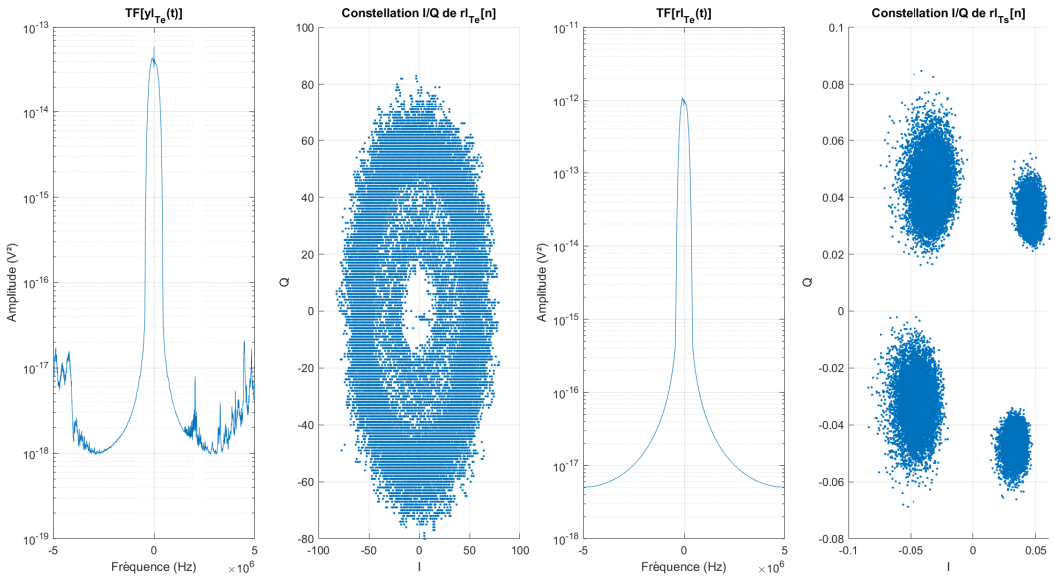

FIGURE 25 – *Transformée de Fourier discrète du signal et sa constellation au récepteur*

5 Conclusion

Ainsi, nous avons établi une chaîne de communications numériques utilisant une modulation QPSK avec ou sans mode différentiel pour un filtre porte ou un filtre en racine de cosinus surélevé. Nous remarquons que, pour le nombre de bits que nous devons traiter, une modulation avec le mode différentiel et l'utilisation d'un filtre en rcos sont à privilégier. Pour corriger le problème lié au matériel, nous avons tenter de mettre en place la détection de plusieurs préambules.

A Scripts MATLAB

A.1 Émetteur pi/4-DQPSK ou QPSK avec pré-synchronisation temporelle et fréquentielle

```
%% Emmetteur pi/4-DQPSK/QPSK avec pre-synchronisation temporelle et frequentielle
close all;
clear;
%% Parametres QPSK
Te=1e-7; % Temps d'echantillonnage des CNA et CAN
Ts=2e-6; % Temps symbole
nb = 2; % Nombre de bits/symbole
M=2^nb; % Nombre de symboles de la constellation
fe=1/Te; % Frequence d'echantillonnage
Ds=1/Ts; % Debit symbole
Db=nb*Ds; % Debit binaire
Fse=floor(Ts/Te); % Facteur de sous-echantillonnage
f0=2.45e9; % Porteuse pour le calcul du PL
roll_off=0.8;
Nfft=2048; % Nombre de points sur lequel la FFT est effectuee
A=127; % Amplitude
Nb_paquets=1;

%% Parametres d'emission
ModeDiff = true; % Si 1 mode differentiel ON, si 0 mode differentiel OFF : QPSK classique
Filtre="rcos"; % Choix du filtre : "porte" ou "rcos"

%% Construction du signal de synchro
preambule = [ones(1,Fse) zeros(1,Fse) ones(1,Fse) zeros(1,2*Fse) ones(1,Fse) zeros(1,Fse) ...
    ones(1,Fse) zeros(1,4*Fse-1)];
preambule = repmat(preambule,1,2); % repmat() : repete le motif de la matrice preambule sur ...
    une fois en ligne et deux fois en colonne

%% Emetteur
fileID = fopen('background.jpg','r');
sb_Te=fread(fileID,'ubit1'); % Recuperation du flux binaire
sb_Te=sb_Te';
fclose(fileID);
Nb = length(sb_Te); % Nombre de bits emis
sb_paquets=reshape(sb_Te,[],Nb_paquets);

sl_Te=0;
sl_Te_test=0;
for j=1:Nb/length(sb_paquets)
    disp(j);
    %% Modulateur numerique
    if ModeDiff
        thetaInit = pi/4; % Theta initial pour le mode differentiel
        ssd_Ts(1) = exp(1i*thetaInit); % Premier symbole differentiel
        ss_Ts=zeros(1,Nb/2/Nb_paquets);
        ss_Ts_1 = exp(1i*thetaInit); % Premier symbole pilote pour correction de phase donc ...
            s_b(1:2) = [0 0]
        ssd_Ts(2)=ssd_Ts(1)*ss_Ts(1); % Creation du symbole differentiel

        for i=1:Nb/2/Nb_paquets % bits vers symbole mode diff
            if sb_paquets((i-1)*2+1:i*2,j)==[0;0]
                ss_Ts(i)=exp(1i*pi/4);
            elseif sb_paquets((i-1)*2+1:i*2,j)==[0;1]
                ss_Ts(i)=exp(1i*3*pi/4);
            elseif sb_paquets((i-1)*2+1:i*2,j)==[1;1]
                ss_Ts(i)=exp(1i*5*pi/4);
            elseif sb_paquets((i-1)*2+1:i*2,j)==[1;0]
                ss_Ts(i)=exp(1i*7*pi/4);
            end
            ssd_Ts(i+1)=ssd_Ts(i)*ss_Ts(i); % Creation du symbole differentiel au rang i+1
        end
        ssd_Ts =[ss_Ts_1,ssd_Ts];
        ss_Te = upsample(ssd_Ts,Fse); % Surechantillonnage de s_sd au rythme Fse
    else % bits vers symbole sans mode diff
        for i=1:Nb/2/Nb_paquets
            if sb_paquets((i-1)*2+1:i*2,j)== [0;0]
                ss_Ts(i)=exp(1i*pi/4);
            elseif sb_paquets((i-1)*2+1:i*2,j)== [0;1]
                ss_Ts(i)=exp(1i*3*pi/4);
```

```
            elseif sb_paquets((i-1)*2+1:i*2,j)== [1;1]
                ss_Ts(i)=exp(1i*5*pi/4);
            elseif sb_paquets((i-1)*2+1:i*2,j)==[1;0]
                ss_Ts(i)=exp(1i*7*pi/4);
            end
        end
        ss_Te = upsample(ss_Ts,Fse); % Surechantillonnage de s_s au rythme Fse
    end

    %% Filtre de mise en forme
    if (Filtre=="porte")
        g = ones(1,Fse); % Version discretre du filtre g(t) echantillonne a Te; elle contient ...
            donc Ts/Te echantillons tous egaux (ici a 1)
    elseif (Filtre=="rcos")
        g=rcosdesign(roll_off,2*M, Fse, 'sqrt');
    end

    sl_Te_conv = conv(ss_Te,g); % Attention lors d'un filtrage numerique, les deux vecteurs a ...
        filtrer doivent etre sur le meme rythme d'echantillonnage !
            if(j==1)
            temp=sl_Te_conv;
        end
    %% Insertion du preambule et scalling
    if ModeDiff
        sl_Te = [sl_Te,preambule*100,sl_Te_conv.*A/max(abs(sl_Te_conv))]; % Multiplication ...
            par A pour pouvoir profiter de l'amplification maximale disponible
    else
        sl_Te = [sl_Te, preambule*A,sl_Te_conv.*A*sqrt(2)/max(abs(sl_Te_conv))];
    end
end

%% Adaptation a la radiologicielle
sl_Te=sl_Te(2:end);
yl_I=real(sl_Te);
yl_Q=imag(sl_Te);

if ModeDiff
    yl_Tx=zeros(1,2*length(sl_Te));
else
    yl_Tx=zeros(1,2*length(sl_Te));
end
yl_Tx(1:2:end)=yl_I;
yl_Tx(2:2:end)=yl_Q;

%% Enrengistrement de la sequence emise
save("sb.mat","sb_Te");
fidID=fopen('QPSK_Tx.raw','w');
fwrite(fidID,int8(yl_Tx),'int8'); % Sauvegarde des echantillons modules
fclose(fidID);

%% Verification de la nature des signaux
% comparaison en DSP theorique et DSP de welch
figure (1);
subplot 121
[DSP_Welch, f] = pwelch(sl_Te(length(preambule):end), ones(1,Nfft), 0, Nfft, fe, 'centered');
if (Filtre=="porte")
    DSP_theo = 2*A^2/Ds*sinc(f/Ds).*sinc(f/Ds);
elseif (Filtre=="rcos")
    DSP_theo=zeros(size(f));
    for i=1:length(f)
        if abs(f(i))<(1-roll_off)/(2*Ts)
            DSP_theo(i)=10*2*A^2/(Ds*Fse);
        elseif abs(f(i))<(1+roll_off)/(2*Ts)
            DSP_theo(i)=10*A^2*(1+cos(pi*Ts/roll_off*(abs(f(i))-(1-roll_off)/(2*Ts))))/(Ds*Fse); ...
                %s'obtient avec une transformation bilineaire
        end
    end
end
semilogy(f,DSP_Welch,f,DSP_theo);
grid on;
title('TF[sl_{Te}(t)] en fonction de la frequence')
xlabel('Frequence (Hz)')
legend('Methode de Welch','DSP theorique')
ylabel ('Amplitude (V^2)')
xlim([-fe/2 fe/2]);
ylim([10^(-5) 1]);
```

```
139
140 subplot 122
141 scatter(real(sl_Te),...
142         imag(sl_Te),'o');
143 grid on
144 title('Constellation I/Q de sl_{Te}[n]')
145 xlabel('I')
146 ylabel ('Q')
147
148 % Diagramme de l'oeil
149 eyediagram(sl_Te(length(preambule):50000+length(preambule)+1),2*Fse,2*Ts);
```

A.2 Récepteur pi/4-DQPSK ou QPSK avec pré-synchronisation temporelle et fréquentielle

```
1  %% Recepteur pi/4-DQPSK/QPSK avec pre-synchronisation temporelle et frequentielle
2  close all;
3  clear;
4  %% Parametres
5  Te=1e-7; % temps d'echantillonnage des CNA et CAN
6  Ts=2e-6; % temps symbole
7  nb = 2; % nombre de bits/symbole
8  fe=1/Te; % Frequence d'echantillonnage
9  Ds=1/Ts; % debit symbole
10 Db=nb*Ds; % debit binaire
11 M=2^nb; % Nombre de symboles de la constellation
12 Nb=3170944;  % Nombre de bits emis %896
13 Fse=floor(Ts/Te); % facteur de sous-echantillonnage
14 f0=2.45e9; % porteuse pour le calcul du PL
15 seuil=0.7;
16 Ptx = 1; % Puissance d'emission (W)
17 roll_off=0.8;
18 A=127; % Amplitude
19 Nb_paquets=1;
20
21 %% Modes de reception
22 ModeDifferentiel=true; % Si 1 mode differentiel ON, si 0 mode differentiel OFF : QPSK classique
23 Filtre="rcos"; % Choix du filtre : "porte" ou "rcos"
24
25 if ModeDifferentiel
26     thetaInit = pi/4; % Theta initial pour le mode differentiel
27     ssd_Ts_1 = exp(1i*thetaInit); % premier symbole pilote pour correction de phase donc ...
           s_b(1:2) = [0 0]
28     Nfft=4096;
29 else
30     Nfft=32768;
31 end
32
33 %% Preambule
34 preambule = [ones(1,Fse) zeros(1,Fse) ones(1,Fse) zeros(1,2*Fse) ones(1,Fse) zeros(1,Fse) ...
       ones(1,Fse) zeros(1,4*Fse-1)];
35 preambule = repmat(preambule,1,2); % repmat() : repete le motif de la matrice preambule sur ...
       une fois en ligne et deux fois en colonne
36 Ep = sum(preambule.^2); %Energie de la sequence de preambule
37
38 %% Filtre adapte
39 if (Filtre=="porte")
40     g = ones(1,Fse); % Version discretre du filtre g(t) echantillonne a Te; elle contient ...
           donc Ts/Te echantillons tous egaux (ici a 1)
41 elseif (Filtre=="rcos")
42     g=rcosdesign(roll_off,2*M, Fse, 'sqrt');
43 end
44 ga = conj(fliplr(g)); % ga(t) = g(Ts-t) (le retard de Ts est implicite en numerique car on ...
       travaille avec des vecteurs causaux)
45
46 %% Acquisition des donnees du recepteur
47 fileID = fopen('QPSK_Tx.raw','r');
48 rl=fread(fileID,'int8=>double'); % Recuperation des echantillons
49 fclose(fileID);
50
51 rl_I=rl(1:2:end);
52 rl_Q=rl(2:2:end);
53 yl_Te=transpose(rl_I+1j*rl_Q);
```

```
s=yl_Te;

%% Canal ( a mettre en simulation et non test)
k = 1.38e-23; % Cte Boltzmann
T = 273.15 + 25; % Temperature bruit du recepteur
N0 = k*T;
varb = N0/2; % Variance du bruit
tau0=1000*rand;
tau0=floor(tau0); % Retard du a la propagation
alpha0 = sqrt(1e-8);  % Coefficient d'attenuation du canal
h_l = 1*[zeros(1,tau0) 1]; % Dirac decale de tau0*Te
Ps = 1/Te*mean(abs(s).^2); % Puissance du signal
yl_Te = sqrt(Ptx/Ps)*yl_Te/127; % Redescente en puissance
y = conv(yl_Te,h_l); % signal en sortie du canal

%% Recepteur (a mettre en simulation et non en test)
Deltaf = 50e-6*f0*2*(rand-0.5); % erreur de frequence Oscillateur Local precis a +-50ppm , ...
    2*(rand-0.5)=nombre aleatoire dans l'intervalle [-1,1]
n = sqrt(varb/2)*(randn(1,length(y))+1i*randn(1,length(y))); % bruit
yl = y.*exp(1i*2*pi*[0:length(y)-1]*Te*Deltaf)+n; % ajout du bruit et de la frequence de decalage
yl_Te=yl;

%% Recepteur (a mettre en simulation et en test)
yl_Te = yl_Te./norm(yl_Te); %% normalisation

%% Pre-synchronisation temporelle
rho=filter(fliplr(preambule),1,abs(yl_Te))./(sqrt(Ep)*sqrt(...
    filter(ones(1,length(preambule)),1,abs(yl_Te).^2))); % Fonction d'intercorrelation

[valMaxk,indk]=maxk(rho,Nb_paquets); % Recuperation de la valeur max et de son indice pour ...
    detecter le preambule
indk=sort(indk);
rl_Ts_conc=0;
for k=1:Nb_paquets
    k
    if valMaxk(k) > seuil
        indSynch = indk(k);
        if ModeDifferentiel
            if (indSynch+1+(Nb/2/Nb_paquets+2)*Fse+length(ga)-1)>length(yl_Te)
                yl_Te_synch = yl_Te(indSynch:end); % Suppression preambule
            else
                yl_Te_synch = yl_Te(indSynch:indSynch+1+(Nb/2/Nb_paquets+2)*Fse+length(ga)-1);
            end
        else
            if (indSynch+Nb/2/Nb_paquets*Fse+length(ga)-1)>length(yl_Te)
                yl_Te_synch = yl_Te(indSynch:end); % Suppression preambule
            else
                yl_Te_synch = yl_Te(indSynch:indSynch+Nb/2/Nb_paquets*Fse+length(ga)-1);
            end
        end
    else
        yl_Te_synch = yl_Te(length(preambule):end);
    end

    %% Filtre adapte
    rl_Te = conv(yl_Te_synch,ga);  % convolution par le filtre adpate

    %% Selection du message utile
    if ModeDifferentiel
        [Rl_Te, f] = pwelch(rl_Te.^8, ones(1,Nfft), 0, Nfft, fe, 'centered');
        Rl_Te_dt=diff(Rl_Te);
        [pks,locs] = findpeaks(Rl_Te_dt,'SortStr','descend');
        if (Filtre=="porte")
            f0_8_esti=f(locs(1)+1)/8; % Estimation frequence residuelle
        elseif (Filtre=="rcos")
            f0_8_esti=f(locs(3)+1)/8; % Estimation frequence residuelle
        end
        rl_Te=exp(-1j*(2*pi*f0_8_esti*(0:length(rl_Te)-1)*Te)).*rl_Te; % Suppression ...
            frequence residuelle

        rl_Ts=rl_Te(length(ga)+1:Fse:length(ga)+(Nb/2/Nb_paquets+1)*Fse+1); % sous ...
            echantillonnage au rythme Fse (correspond a un echantillonnage de r_l(t) au ...
            rythme Ts)
        rl_Ts_1=rl_Ts(1);
        rl_Ts_2=rl_Ts(2);
        rl_Ts = rl_Ts(3:end)./(rl_Ts(2:end-1));
```

```
125
126         % Pre-synchronisation frequentielle tant que 2pi*Nb*f_erreur<pi/4
127         Phase_correction = angle(ssd_Ts_1)-angle(rl_Ts(2))+pi; % correction de phase initiale
128         rl_Ts = rl_Ts.*exp(1j*Phase_correction);
129         rl_Ts=rl_Ts./max(abs(rl_Ts));
130     else
131         [Rl_Te, f] = pwelch(rl_Te.^4, ones(1,Nfft), 0, Nfft, fe, 'centered');
132         [val_max,arg_max]=max(abs(Rl_Te));
133         f0_4_esti=f(arg_max)/4; % Estimation frequence residuelle
134         rl_Te=exp(-1j*(2*pi*f0_4_esti*(0:length(rl_Te)-1)*Te)).*rl_Te; % Suppression ...
                frequence residuelle
135         rl_Ts=rl_Te(length(ga)+1:Fse:length(ga)+(Nb/2/Nb_paquets-1)*Fse+1); % sous ...
                echantillonnage au rythme Fse (correspond a un echantillonnage de r_l(t) au ...
                rythme Ts)
136         % Pre-synchronisation frequeltielle avec f_residuelle<fe/(2Nfft)
137     end
138     figure(3), plot(rl_Ts,'*');
139     rl_Ts_conc=[rl_Ts_conc, rl_Ts];
140 end
141
142 %% Demodulateur numerique
143 rl_Ts_conc=rl_Ts_conc(2:end);
144 sb_esti = zeros(1,Nb);
145 sb_esti(2:2:end) = real(rl_Ts_conc)<0;
146 sb_esti(1:2:end) = imag(rl_Ts_conc)<0;
147
148 %% BER
149 sb=load("sb.mat","sb_Te");
150 sb=sb.sb_Te;
151 [n,r]=biterr(sb,sb_esti) % Calcul du taux d'erreur binaire avec n le nombre de bits faux et r ...
        le ratio
152
153 %% Enrengistrement
154 fidID=fopen('QPSK_Rx.jpg','w');
155 fwrite(fidID,int8(sb_esti),'ubit1');
156 fclose(fidID);
157
158 %% Verification de la nature des signaux recus
159 % comparaison en DSP theorique et DSP de Welch
160 figure (1);
161 subplot 141
162 [DSP_yl_Te, f] = pwelch(yl_Te(length(preambule):end), ones(1,Nfft), 0, Nfft, fe, 'centered');
163 semilogy(f,DSP_yl_Te);
164 grid on;
165 title('TF[yl_{Te}(t)] en fonction de la frequence')
166 xlabel('Frequence (Hz)')
167 ylabel ('Amplitude (V^2)')
168
169
170 % Constellation I/Q
171 subplot 142
172 scatter(real(s),...
173         imag(s),'.');
174 grid on
175 title('Constellation I/Q de yl_{Te}[n]')
176 xlabel('I')
177 ylabel ('Q')
178 subplot 144
179 scatter(real(rl_Ts_conc),...
180         imag(rl_Ts_conc),'o');
181 grid on
182 title('Constellation I/Q de r_{l}[n]')
183 xlabel('I')
184 ylabel ('Q')
185
186 figure (1);
187 subplot 143
188 [DSP_rl_Te, f] = pwelch(rl_Te(length(preambule):end), ones(1,Nfft), 0, Nfft, fe, 'centered');
189 semilogy(f,DSP_rl_Te);
190 grid on;
191 title('TF[rl_{Te}(t)]en fonction de la frequence')
192 xlabel('Frequence (Hz)')
193 ylabel ('Amplitude (V^2)')
194
195 figure,plot(rho(1:indSynch*2));
196
```

```
% Diagramme de l'oeil
eyediagram(rl_Te(1:50000),2*Fse,2*Ts);
```

A.3 Réseaux des SNR en témpérature en fonction de la distance

```
%% Concevoir une chaine de communication numerique QPSK avec synchro sur preambule
clear;
close all;
clc;

%% Initialisation
Ts=2e-6; % temps symbole
nb = 2; % nombre de bits/symbole
Ds=1/Ts; % debit symbole
Db=nb*Ds; % debit binaire
f0=1.2e9; % porteuse pour le calcul du PL

%% Variables
T=[-50:5:50]; %[C]

for i=1:length(T)
    %% Emetteur
    Ptx = 1; % Puissance d'emission (W)
    Ptx_dBm = 10*log10(Ptx)+30;

    Gtx_dBi=8;
    Gtx = 10^(Gtx_dBi/10);

    %% Recepteur
    Grx = 1;
    Grx_dBi=10*log10(Grx);

    %% bruit ambiant
    k = 1.38e-23; % Cte Boltzmann
    T_i= 273.15 + T(i); % Temperature bruit du recepteur
    N0 = k*T_i; %[W/Hz]
    Pbruit = Db*N0; %[W]
    Pbruit_dBm = 10*log10(Pbruit)+30;

    %% Pertes par espace libre
    c=3e8; % vitesse de la lumiere
    R = [100:100:1000,2000:1000:10000,20e3:10e3:300e3];
    PL = ((f0*4*pi*R)./c).^2; % Path Loss
    PL_dB= 10*log10(PL);

    %% Calcul du SNR en fonction de R et T
    RSB_dBm=Ptx_dBm+Gtx_dBi+Grx_dBi-PL_dB-Pbruit_dBm;
    EbN0_dBm = RSB_dBm;

    figure(1)
    semilogx(R,RSB_dBm)
    xlabel('Distance (m)');
    ylabel('SNR (dB)');
    grid on;
    hold on;
end
title('Reseaux des SNR en temperature en fonction de la distance');
xlim([0 300e3]);
hold off;
```