Modélisation d'une chaîne OFDM

Préambule

Le but de cette séance est de modéliser une chaîne de transmission OFDM selon le standard WLAN 802.11a sous Python. Pour cela, il nous faut modéliser :

- l'émetteur en bande de base avec étude du PAPR,
- le récepteur et ajout d'un canal pour le calcul de l'EVM (Error Vector Magnitude) et le TEB.

Copier le répertoire *tpOFDM* dans votre espace de travail. Il contient les fichiers suivants :

- *ofdmTranceiver.py* : fichier contenant les paramètres de simulation,
- *commonFunction.py* : fichier contenant des fonctions indispensables : *bitMapping()*, *ifftAddIg()*, *removeIGandFFT()*, *demapping2bit()*, ...
- La fonction *plotSpectrum()* est déjà disponible pour tracer le spectre.

1 Chaîne d'émission

Voici la chaîne d'émission que nous allons modéliser. A noter l'absence des 4 pilotes et les 12 sous-porteuses nulles pour ce TP.

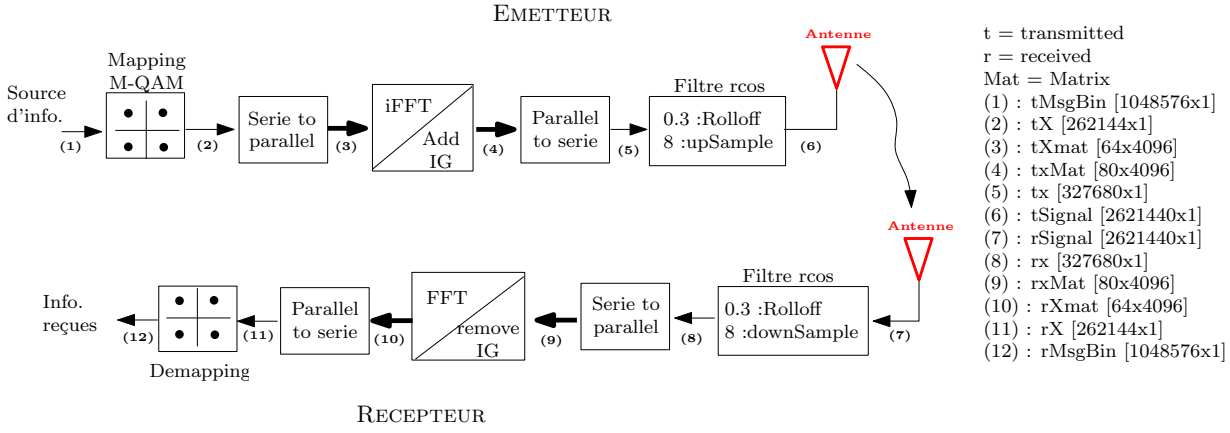

Figure 1 – Système OFDM classique

Pour simplifier la vérification du code, vous donnerez les mêmes noms aux variables Python que ceux qui sont dans le schéma. Si vous en créez d'autres, vous les appellerez en cohérence avec leur signification physique. Pensez toujours à vérifier les tailles des vecteurs

Ouvrez le fichier *ofdmTranceiver.py* (sous VSC par exemple). Ce fichier donne la structure du programme avec les indispensables commentaires. En préambule, vous avez les paramètres de simulation. Avec ces paramètres, les dimensions des variables sont celles qui sont sur la figure ci-dessus (ça vous permettra de vérifier vos dimensions au fur et à mesure). Il faut réaliser des programmes qui sont les plus génériques possibles pour permettre des vérifications en jouant sur les paramètres du système OFDM.

Votre travail est de réaliser les étapes suivantes (A chaque étape, vérifiez bien vos résultats puis appelez le professeur) :

- écrire la fonction *bitMapping()* (du point-1 au point-3 sur le schéma) qui génère le signal binaire, réalise le mapping M-QAM et place les symboles mappés sous forme de matrice tXmat.
- écrire à présent *ifftAddIg()* qui permet de passer en temporel (pt-3 à 5). Elle réalise l'iFFT puis rajoute les échantillons de l'intervalle de garde selon la méthode du Préfixe Cyclique (CP).
- écrire à présent *rrcos()* qui sur-échantillonne puis calcule et applique un filtre d'émission (filtre en cosinus surélevé de facteur de Rolloff égal à 0.3 avec un facteur de sur-échantillonnage de 8),

1. Calculez le PAPR en dB du signal transmis (pt-6). Faites plusieurs simulations, normalement le PAPR est entre 11 dB et 12 dB.

2. Faites varier la longueur N de l'iFFT (*nFFTSize* = 16, 32, 64, 128, 256, 512) et prélevez le PAPR linéaire (pas en dB). Tracez le PAPR linéaire en fonction de $\sqrt{3\,N}$. Que constatez-vous ?
3. On fixe $N = 64$ sous-porteuses. Faites varier le nombre d'état M-qam du mapping (*M_qam* = 4, 8, 16, 32, 64, 128, 256, 512, 1024). Tracez le PAPR linéaire en fonction de M. Que constatez-vous ?
4. Faites de même en fixant à présent $N = 128$ pour s'assurer de votre résultat, puis conclure.
5. En utilisant la fonction *toPlotSpectrum()* avec un échantillonnage unitaire, tracez le spectre du signal transmis *tSignal*. Relevez la largeur de la bande occupée ? Justifiez ce résultat.

Passage rapide en RF pour bien comprendre : Jusqu'à présent, nous étions en bande de base, ce qui donne des signaux complexes. Vous vous doutez bien que l'amplificateur de puissance et l'antenne ne transmettent pas de signaux complexes !! Dans la vraie vie, le signal est RF (modulé par une porteuse RF) et il est bien sûr réel (transmission analogique).

1. Donnez l'expression du signal RF transmis en fonction de l'enveloppe complexe $\tilde{v}_e(t) = I(t) + j\,Q(t)$ et la porteuse. On notera f_c la fréquence porteuse,
2. Reprenez votre programme précédent et générer la porteuse en prenant une fréquence normalisée $f_c = 0.1$ [1]. Modulez le signal en bande de base par cette porteuse pour générer le signal réellement transmis. Visualisez-le et comparer-le (faire un zoom) avec la valeur absolue du signal en bande de base *tSignal*. Justifiez en deux lignes.
3. Calculez son PAPR et comparer-le avec celui en bande de base. Normalement, il est environ $3\,dB$ plus grand.

2 Chaîne de réception

Dans cette partie, on va revenir en bande de base. Faites un copie de *ofdmTranceiver.py* où cas où puis mettre la modulation RF en commentaire.

Le récepteur est le dual de l'émetteur : Votre travail est de réaliser les étapes suivantes :

- insérer un canal AWGN avec avec un SNR de 30 dB (pt-6 à 7) (faible bruit).
- appliquez le filtre de réception avec le même Rolloff qu'à l'émission et un sous-échantillonnage de 8 (pt-7 à 8). Pour vérifier, comparez les parties réelles des signaux *tx* et *rx*. Vous allez voir s'il y a un retard. Le corriger si besoin en alignant les signaux. Vérifiez bien que les signaux sont identique après synchronisation.
- écrire la fonction nommée *removeIGandFFT()* qui supprime le CP et effectue la FFT sur chaque symbole OFDM (pt-8 à 10). Vérifiez que les parties réelles et imaginaires du premier symbole de *tXmat(:,1)* et *rXmat(:,1)* sont semblables.
- écrire la fonction *demapping2bit()* : démodulation M-QAM (pt-10 à 12). Vérifiez que le message binaire à la réception et semblable à celui de l'émission. Tracez la différence entre les deux messages binaires pour voir si elle est nulle.

On va tracer quelques critères comme l'EVM et le TEB :

1. Faites varier le SNR entre 2 et 16 dB avec un pas de 1 et tracez la courbe du TEB en fonction du SNR. Que constatez-vous ?
2. Mettre une modulation 32-QAM puis 64-QAM et superposez les 3 courbes du TEB. Conclure.
3. Allez sur internet et documentera vous rapidement sur l'EVM. Résumez ce que vous avez compris.
4. Ecrire la fonction *calculateEVM()*. Faites varier le SNR entre 2 et 16 dB et tracez la courbe de l'EVM en fonction du SNR. Que constatez-vous ?
5. Idem pour une modulation 256-SAM. Superposez les courbes puis justifiez ce résultat.

Pour ceux qui vont vite : Insérer un canal sélectif de type Rayleigh :

1. Comparer avec les résultats de l'EVM et du TEB précédents puis conclure.
2. Faites varier le nombre d'échantillons de l'IG (0, 2, 4 puis 16) pour un SNR égal 30 dB (faible bruit). Que peut-on conclure ?

1. Voici pour comprendre d'où vient cette fréquence normalisée de 0.1 : Par défaut, Matlab travaille à la fréquence d'échantillonnage $f_s = 1$. Ainsi, les quantités représentant des fréquences (ou des débits) devront être normalisées par rapport à f_s. Prendre $f_c = 0.1$ revient donc à utiliser une porteuse dont la fréquence réelle est 1/10 de f_s. Exemple : Si $f_c = 2\,GHz$, on utilisera une fréquence d'échantillonnage 10 fois plus grande, donc $f_s = 20\,GHz$, pour respecter la limite de Shannon