TP Mineure Signal S7
Cours de Communications Numériques

2ème année ENSEA

Octobre 2023

Les TPs de la Mineure Signal sont réalisés avec le logiciel Matlab. Votre présence est obligatoire et toute absence non-justifiée engendre une diminution de note. Les étudiants qui ne peuvent être présents à cause des raisons médicales, doivent contacter le.a professeur.e encadrant rapidement et suivre le protocole spécifique qui leur sera indiqué.

Vous devez valider 3 check-points par séance, avec le.a professeur.e encadrant qui les notera.

A la fin de chaque séance et avant de quitter le poste de travail, chaque binôme a l'obligation d'envoyer à son encadrant.e un mail avec un ficier `*.pdf` contenant **le compte-rendu de TP** : formules et calculs (peuvent être rendus sur papier en séance), courbes anotées (limitées à 8 par séance), avec une attention particulière à l'analyse et l'interprétation des résultats.

Concernant les codes MatLab : suivez les instructions de votre professeur.e encadrant.

Hint 1 Matlab : Utilisez `help nomfonction` *dans la console pour obtenir les règles d'utilisation des différentes fonctions (ici* `nomfonction.m`*).*
Hint 2 Matlab : pour créer la fonction `myfunction.m` *(une seule fonction par fichier* `*.m`*), utiliser la syntaxe*

```
function [output1, output2,...]  = myfunction(input1, input2,...)
```

et la sauvegarder dans le fichier `myfunction.m`*.*
Hint 3 Matlab : vous devez TESTER toutes vos fonctions crées dans un ficher script à part, e.g., `main.m` *(qui ne contient pas des définitions des fonctions mais qui fait appel aux fonctions déjà définies) afin de assurer leur comportement conforme.*

THEME 1 : Communication en bande de base BPSK

Les trois points à valider pour le Thème 2 sont les suivants.

- **Relation entre le rapport signal à bruit (RSB) et le RSB normalisé : $\frac{E_b}{N_0}$.**
- **Canal idéal : comparaison entre le taux d'erreur binaire (simulateur) et la courbe théorique.**
- **Canal non-idéal : visualisation de la constellation BPSK qui subit des IES, $x_n,\ n = 0, \ldots, N-1$.**

Remarques et astuces

Notation : nous allons utiliser la notation des signaux discrètes du module Communications Numériques telle que $x_n,\ n = 0, \ldots, N-1$ représente le n-éme échantillon.

Fonctions MatLab utiles : `rand`, `randn`, `sign`, `filter`, `plot`, `xlabel`, `ylabel`, `title`, `erfc`, `find`, `length`, `semilogy`, *etc.*

*Hint : Pour tracer le taux d'erreur nous utilisons la méthode de Monte-Carlo : à savoir la simulation d'*un grand nombre *N de transmissions binaires. Parmi les N symboles transmis nous calculons la fréquence empirique des occurrences d'erreur (comparaison entre a_n et $\widehat{a}_n$).*

■

1 Canal idéal de Nyquist AWGN

Par simplicité, nous allons considérer dans cette première partie un canal idéal de Nyquist avec un bruit additif Gaussien (AWGN), illustré dans la Figure 1. L'objectif sera l'étude de la probabilité d'erreur binaire à la réception.

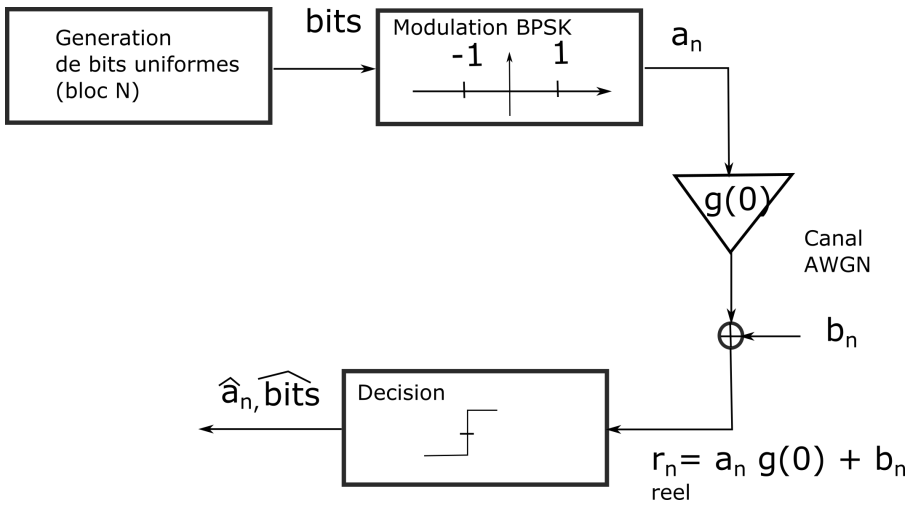


Figure 1: Chaîne de communication en bande de base d'un signal BPSK à travers un canal idéal de Nyquist.

♢ Chaîne de communication BPSK

• Écrire une fonction `genBPSK.m` avec pour entrée N et sortie un vecteur de N échantillons du signal modulé BPSK : a_n, $n = 0, \ldots, N-1$. Vous pouvez utiliser `rand.m` pour créer une séquence aléatoire des valeurs entre 0 et 1. Puis, soustrayez -0.5 et prenez-en le signe à l'aide `sign.m`[1].

• Créer une fonction `genBruitN.m` avec pour entrée une valeur σ_b^2 et pour sortie une séquence de N échantillons du bruit b_n, $n = 0, \ldots, N-1$ aléatoires suivant une loi Gaussienne de moyenne nulle et variance σ_b^2. Vous pouvez utiliser la fonction `randn.m`[2].

• Afin de coder le bloc de décision, créer une fonction `decodeBPSK.m` avec pour entrée un vecteur r_n de N échantillons réels observés à la réception et pour sortie un vecteur $\widehat{a}_n$ de N échantillons dans $\{-1, +1\}$, qui représentent les estimations des symboles BPSK transmis (a_n, $n = 0, \ldots, N-1$).

• Dans le programme principal, simulez la chaîne de communication pour $N = 512$, $\sigma_b^2 = 0.5$ et $g(0) = 1$. Visualisez le signal modulé BPSK, a_n, ainsi que le signal estimé, $\hat{a}_n$. Quel est le taux d'erreur binaire à la réception, i.e., le ratio entre le nombre de symboles erronés vs. le nombre de symboles transmis (N) ?

♢ Taux d'erreur binaire du canal AWGN

• Relier le rapport signal à bruit RSB $= \frac{g(0)^2}{\sigma_b^2}$ au rapport signal à bruit normalisé $\frac{E_b}{N_0}$.

• Pour chaque valeur $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots 12$ **[dB !]**, calculer la variance du bruit σ_b^2 pour $g(0) = 1$ ainsi que le taux d'erreur binaire correspondants pour $N = 512$.

• Tracer la courbe du taux d'erreur binaire en fonction de $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots, 12$ **[dB !]** pour $N = 512$. Commentez.

• Quelle est la valeur de N minimale afin de pouvoir tracer la courbe du taux d'erreur binaire pour toute la plage de $\left[\frac{E_b}{N_0}\right]_{dB}$?

♢ **Check-point 1** **Relation entre le rapport signal à bruit (RSB) et le RSB normalisé : $\frac{E_b}{N_0}$.**

♢ Probabilité d'erreur binaire théorique

• Tracez la courbe de la probabilité d'erreur binaire théorique vue en cours, en fonction du rapport signal à bruit normalisé $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots, 12$ **[dB !]** :

$$P_{b(\min)}(e) = \mathcal{Q}\left(\sqrt{\frac{2E_b}{N_0}}\right).$$

Vous pouvez utiliser la fonction MatLab `erfc.m` par example, en faisant attention à la relation avec la fonction $\mathcal{Q}(x) \neq \text{erfc}(x)$.

♢ Comparaison des courbes d'erreur : pratique et théorique

• Comparer la courbe pratique (obtenue via simulation) avec la courbe théorique. Commentez.

• Quel est le SNR (en dB) qui permet de garantir pour une probabilité d'erreur binaire d'environ 10^{-3} ?

♢ **Check-point 2** **Canal idéal : comparaison entre le taux d'erreur binaire (simulateur) et la courbe théorique.**

[1] Afin de vérifier le comportement correct de `genBPSK.m`, tracer l'histogramme de la sortie.

[2] Afin de vérifier le comportement correct de `genBruitN.m`, tracer l'histogramme de la sortie.

2 Interférence entre symboles (IES) et égalisation

Dans cette partie, nous allons considérer un canal convolutif qui introduit des IES comme illustré dans la Figure 2. Nous allons étudier une méthode simple d'égalisation, qui permet de diminuer ou annuler les effets des IES à la réception.

♢ Canal non-idéal

• Créer `genSig.m` avec pour entrée N et sortie un vecteur de N échantillons du signal signal utile x_n, $n = 0, \ldots, N-1$. Utilisez la fonction `filter.m` pour filtrer le signal BPSK $a_n,\ n = 0, \ldots, N-1$ par $G(z) = 0.5 + 0.2z^{-1}$ afin d'obtenir le signal utile x_n, $n = 0, ..., N-1$.

• écrire l'expression de la réponse impulsionnelle du filtre global du canal $g_n, \forall n$ à partir de la TZ $G(z)$ ci-dessus. écrire ensuite l'expression de x_n pour $n = 0, 1, 2, 3$ en supposant que $a_n = 0,\ \forall n < 0$.

• Visualiser le signal utile $x_n,\ n = 0, \ldots, N-1$. Commenter.

♢ **Check-point 3** **Canal non-idéal : visualisation de la constellation BPSK qui subit des IES, $x_n,\ n = 0, \ldots, N-1$.**

♢ Taux d'erreur binaire du canal non-idéal

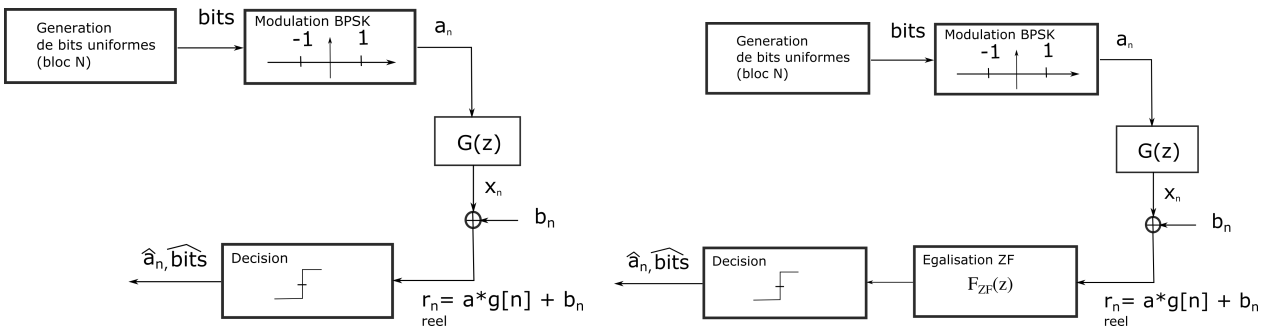
• Utiliser les fonctions développées dans la partie 1 afin de tracer le taux d'erreur binaire de la chaîne de communication de la Figure 2a en fonction de $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots, 12$. Quel est l'effet des IES et du canal non-idéal $G(z)$? Commenter.

♢ Égalisation

• Une méthode simple d'égalisation, appelée forçage à zéro (*zero forcing*), consiste à inverser le filtre $G(z)$ à la réception. Créer une fonction `egaliZF.m` qui a pour entrée la séquence de N symboles reçues $r_n,\ n = 0, \ldots, N-1$ et pour sortie une sequence $t_n,\ n = 0, \ldots, N-1$ obtenue en filtrant l'entrée par $F_{ZF}(z) = G(z)^{-1}$.

• Tracer le taux d'erreur binaire de la chaîne de communication de la Figure 2b en fonction de $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots, 12$. Commenter.

• Quelles sont les inconvénients de cette méthode dans la pratique ?



(a) figure
Sans égalisation.

(b) figure
Avec égalisation.

Figure 2: Chaîne de communication en bande de base d'un signal BPSK à travers un canal non-idéal $G(z)$.

THEME 2 : Communication complexe équivalente en bande de base de la 16-QAM

Les trois points à valider pour le Thème 4 sont les suivants.

- **Visualisation de la 16-QAM (constellation complexe et code de Gray).**
- **Détection dans l'espace complexe de la constellation (test et visualisation).**
- **Comparaison des courbes d'erreur binaire et symbole (pratiques vs. théoriques).**

Remarques et astuces

Notation : nous allons utiliser la notation des signaux discrètes du module Communications Numériques telle que x_n, $n = 0, \ldots, N-1$ *représente le* n*-éme échantillon et* $x_n \equiv x_N[n]$ *(notation TNSII).*

*Hint : Pour tracer le taux d'erreur binaire nous utilisons la méthode de Monte-Carlo : à savoir la simulation d'*un grand nombre N *de transmissions binaires. Parmi les* N *bits transmis nous calculons la fréquence empirique des occurrences d'erreur (comparaison entre* bits_n *et* $\widehat{\mathrm{bits}}_n$*). Le taux d'erreur symbole est calculé d'une manière similaire en comparant les symboles 16-QAM* a_n *avec leur estimations* $\widehat{a}_n$*.*

MatLab : consultez les fonctions `bin2dec`, `dec2bin`, `char`, `find`, `plot` *en complément des fonctions utilisées dans les autres Thèmes.* ■

1 Constellation 16-QAM

L'objectif est de visualiser la constellation et le code de Gray pour la 16-QAM d'une manière similaire à la constellation QPSK dans la Figure 3.

♢ Constellation complexe et code de Gray

• Écrire une fonction `const16QAM.m` qui retourne une matrice complexe, $\mathrm{mComplex}$ de taille 4×4, qui contient toutes les valeurs possibles des symboles $a_n = a_n^I + j a_n^Q$ avec $a_n^I, a_n^Q \in \{\pm 1, \pm 3\}$.

• Écrire une fonction `Gray16QAM.m` qui génère mGray la matrice de taille 4×4 qui contient les codes de Gray (stockés en decimal) correspondants.

Exemple QPSK ($M = 4$)

Les deux matrices de taille 2×2 *de la QPSK (i.e., 4-QAM) illustrée dans la Figure 3 sont:*

$$\mathrm{mComplex} = \begin{pmatrix} -1+j & 1+j \\ -1-j & 1-j \end{pmatrix} \quad \textit{et} \quad \mathrm{mGray} = \begin{pmatrix} 1 & 3 \\ 0 & 2 \end{pmatrix}, \quad \textit{en binaire} \quad \begin{pmatrix} '01' & '11' \\ '00' & '10' \end{pmatrix},$$

où mComplex *est la matrice des symboles complexes* a_k *et* mGray *est la matrice des codes de Gray en decimal.* Attention : *le mapping de Gray n'est pas unique !* ■

♢ Visualisation de la 16-QAM

• Dans le programme principal, générer les deux matrices pour la 16-QAM, initialiser $M = 16$ et visualiser les points de la constellation ainsi que les étiquettes binaires de Gray en utilisant le code MatLab mis à votre disposition.

♢ **Check-point 1** **Visualisation de la 16-QAM (constellation complexe et code de Gray).**

2 Transmission complexe équivalente en bande de base de la 16-QAM

L'objectif ici est d'écrire une fonction qui simule la chaîne de communication de la 16-QAM illustré dans la Figure 4. Pour ceci, chaque bloc sera écrit dans une fonction à part, comme détaillé par la suite.

♢ Émetteur

• Écrire une fonction `genBin.m` qui génère bits_n, $n = 0, \ldots, N-1$, une sequence de N valeurs binaires dans $\{0, 1\}$ uniformément distribués, en utilisant par example la fonction `randi.m`.

• Écrire une fonction `mappingGray.m` qui code la séquence binaire à transmettre, bits_n, dans une séquence des $N/4$ symboles a_n de la 16-AQM en utilisant les deux matrices mComplex et mGray déterminées dans la partie 1. Expliquez pour quoi la sequence des symboles a_n a une taille égale à $N/4$.

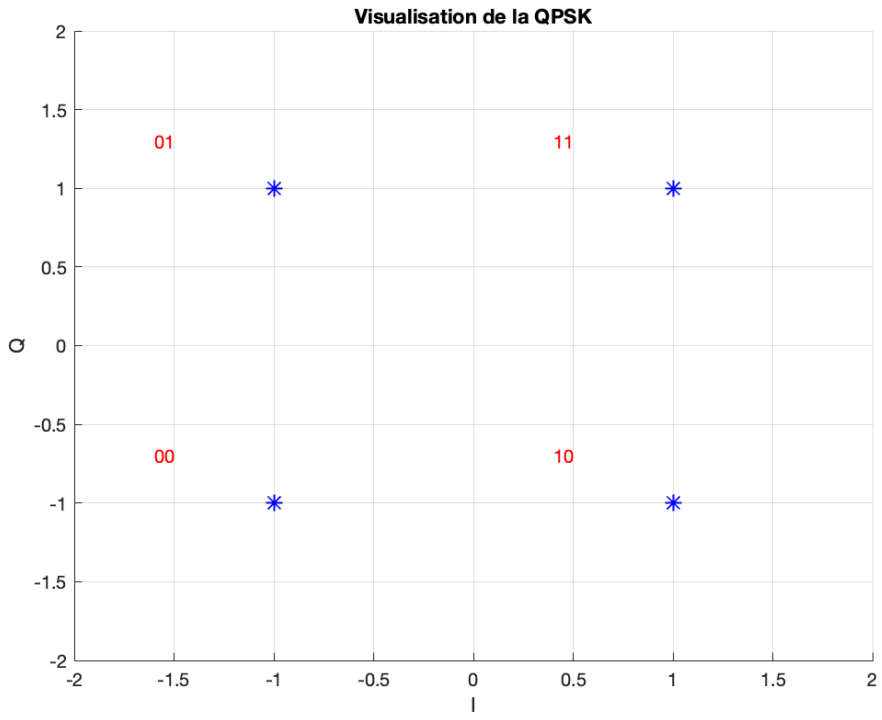


Figure 3: Visualisation de la QPSK (constellation complexe et code de Gray).

Exemple QPSK ($M = 4$)

Pour une QPSK, supposons que la séquence binaire à transmettre ($\mathrm{bits}_n,\ n = 0, \ldots, N$*) est :*
$0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ \ldots$
• *traiter les bits en groupes deux par deux (la taille du groupe est* $\log_2 M$ *où* M *est la taille de la constellation)*
$0,\ 1 \mid 1,\ 0 \mid 0,\ 1 \mid 0,\ 0 \mid 1,\ 1 \mid 1,\ 1 \mid \ldots$
• *les transformer en decimal*
$1, \quad 2, \quad 1, \quad 0, \quad 3, \quad 3, \ldots$
• *utiliser la matrice* mGray *de l'Exemple 1 pour identifier la position* $(\mathrm{lin}, \mathrm{col})$ *de chaque code*
$(1,1);\ (2,2);\ (1,1);\ (2,1);\ (1,2);\ (1,2);\ \ldots$
• *ensuite lire* $\mathrm{mComplex}(\mathrm{lin}, \mathrm{col})$ *(Exemple 1) pour identifier la séquence des symboles QPSK* $a_n,\ n = 0, \ldots, N$ *correspondante :*
$-1 + j,\ 1 - j,\ -1 + j,\ -1 - j,\ 1 + j,\ 1 + j,\ \ldots$

■

Remarques

Contrairement à la BPSK (Thème 2) pour laquelle le mapping de Gray est trivial : $0 \to -1$ *et* $1 \to 1$*; quand* $M > 2$ *le bloc de génération des bits doit être explicitement inclus dans la chaîne de communication. L'information binaire à transmettre est traitée par paquets de taille* $\log_2(M)$*, qui sont codés dans des symboles complexes de la* M*-QAM.* ■

♢ Canal complexe AWGN

• Écrire une fonction `canalAWGN.m` qui a pour entrée la sequence complexe des $N/4$ symboles a_n et la variance du bruit σ_b^2 et pour sortie la sequence de $N/4$ valeurs complexes r_n à l'entrée du récepteur. Le bruit additif b_n est une variable Gaussienne complexe avec la partie réelle et la partie imaginaire indépendantes telles que $b_k^I, b_k^Q \sim \mathcal{N}\left(0, \sigma_b^2\right)$. Par simplicité nous choisissons $g(0) = 1$.
• Dans le programme principal, tester toutes ces fonctions en visualisant les symboles reçues r_n sur la même figure que les symboles de la 16-QAM pour $N = 512$ et $\sigma_b^2 = 0.5$.

♢ Récepteur

• Le bloc de décision prend en entrée la séquence de $N/4$ valeurs complexes r_n et retourne la sequence de $N/4$ symboles estimés $\widehat{a}_n$ de la constellation 16-QAM. Vous pouvez utiliser une méthode par seuil de comparaison (similaire à la BPSK) dans l'espace 2D complexe. Par contre, cette méthode est fastidieuse et le code résultant est complexe. Comme alternative vous pouvez aussi chercher, pour chaque observation r_n, quel est le symbole complexe le plus proche de la 16-QAM. Tester votre méthode en utilisant la visualisation d'un seul symbole reçu ainsi que le symbole estimé correspondant.
• écrire une fonction `demapGray.m` qui permet d'effectuer l'operation inverse à la fonction `mappingGray.m`. Cette fonction va exploiter les deux matrices $\mathrm{mComplex}$ et mGray afin de reconstituer la sequence binaire $\widehat{\mathrm{bits}_n}$ de taille N à partir des $N/4$ symboles estimés $\widehat{a}_n$.

♢ **Check-point 2** **Détection dans l'espace complexe de la constellation (test et visualisation).**

♢ Chaîne de transmission globale

• Écrire une fonction globale `chaine16QAM.m` qui simule la chaîne de transmission de la Figure 4. Cette fonction a pour entrée deux valeurs N, σ_b^2 et $g(0)$; en sortie elle a quatre vecteurs : deux vecteurs bits_n et $\widehat{\text{bits}}_n$ de taille N et deux vecteurs a_n et $\widehat{a}_n$ de taille $N/4$. Testez cette fonction dans le programme principal.

3 Courbes d'erreur de détection

♢ Taux d'erreur binaire et symbole

• Pour une M-QAM : chercher dans les diapos du cours le lien entre le $\text{RSB} = \frac{g(0)}{\sigma_b^2}$ et le SNR normalisé, $\frac{E_b}{N_0}$.

• Pour chaque valeur de $\left[\frac{E_b}{N_0}\right]_{dB} = 0, \ldots, 12$ [dB !] calculer le σ_b^2 pour $g(0) = 1$, ainsi que le taux d'erreur symbole et binaire correspondants de votre simulateur `chaine16QAM.m` de la partie 2. Pour le taux d'erreur symbole, vous devez compter le nombre de symboles différents entre la sequence transmise a_n et la séquence estimée $\widehat{a}_n$. Pour le taux d'erreur binaire, vous devez compter le nombre des bits différents entre la sequence transmise bits_n et la séquence estimée $\widehat{\text{bits}}_n$.

♢ Probabilité d'erreur binaire et symbole (théoriques)

• Écrire la formule de la probabilité d'erreur binaire $P_{b(min)}(e)$ paramétrée par M, la taille de la constellation. Tracer la courbe de la probabilité d'erreur binaire pour $M = 16$.

• Écrire la relation entre la probabilité d'erreur symbole et $P_{b(min)}(e)$ ci-dessus pour un codage de Gray et en supposant un bruit suffisamment petit. Tracer la courbe de la probabilité d'erreur symbole pour $M = 16$.

♢ Analyse et discussions

• Comparer les courbes estimées avec le simulateur et les courbes théoriques (pour les erreurs binaires et symboles). Commenter.

• Comparer la communication en bande de base de la BPSK et la communication en bande transposée de la

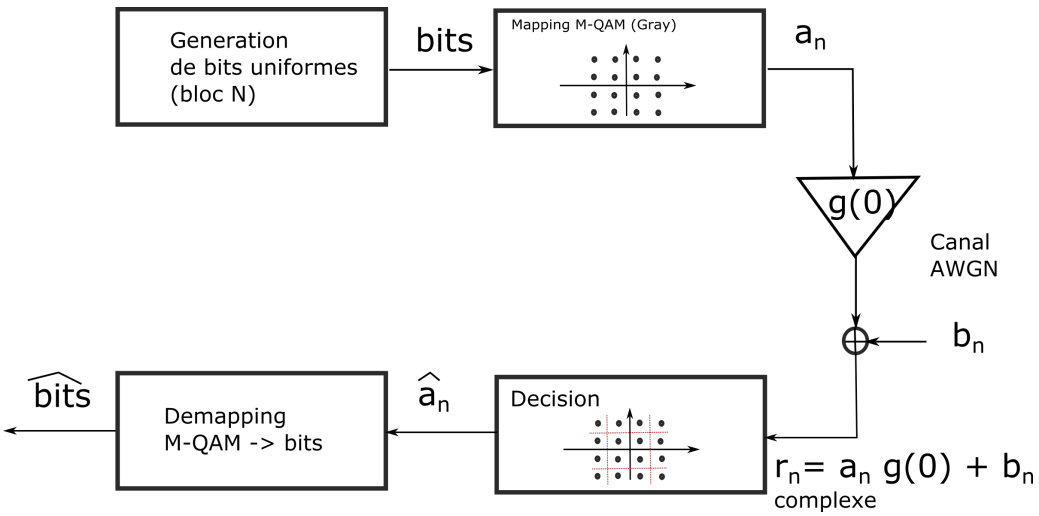


Figure 4: Chaîne de communication de la 16-QAM.

16-QAM en terme du débit binaire, de la probabilité d'erreur binaire et de l'efficacité spectrale. Commenter.

• Si le temps imparti permet, tracer aussi les courbes d'erreur de la QPSK (en adaptant le simulateur au cas $M = 4$) et faire une comparaison complète: BPSK vs. QPSK vs. 16-QAM.

♢ **Check-point 3** **Comparaison des courbes d'erreur binaire et symbole (pratiques vs. théoriques).**

4 Appendice

```
% Visualiser la constellation M-QAM et le codage de Gray
%-------------------------------------------------------
% M : taille de la constellation
% mComplex : matrice contenant les symboles complexes
% de taille (sqrt(M), sqrt(M))
% mGray : matrice contenant les codes de Gray (en decimal)
% de taille (sqrt(M), sqrt(M))

M = 16;
x = real(mComplex(:));
y = imag(mComplex(:));
z = mGray(:);

% new figure
figure

% dessiner les points de la constellation
scatter(x,y,50,'b*');
axis([-sqrt(M) sqrt(M) -sqrt(M) sqrt(M)]);

% ajouter codes de Gray
for k = 1 : M
    text(x(k)-0.6,y(k)+0.3,...
        dec2base(z(k),2,log2(M)),'Color',[1 0 0]);

end

% parametres figure
title('Codage de Gray de la M-QAM');
xlabel('I');
ylabel('Q');
grid on
```