Telecom205-D1

Vincent, Antonin

Février 2024

Contents

Note : les figures sont disponibles au format *fig* sur le dépôt Git dans le dossier `d1/fig` pour plus de détails.

1 D1

1.1 Calculer le générateur polynomial des codes BCH considérés et calculer le rendement du code

On a $GF(32) = GF(2)[x]/1 + x^2 + x^5$. On note α l'élément primitif.

Pour le BCH correcteur de $t = 1$ erreur, il faut donc 2 erreurs consécutives. On sait que $p(x) = 1+x^2+x^5$ s'annule en α et tous les α^i où i est une puissance de 2. Donc $p(x)$ s'annule en α et en α^2. On prend donc $g(x) = p(x)$ pour le polynôme générateur. On a alors un degré de 5 soit $n - k = 5$. Or $N_c = 31$ (d'après l'énoncé). Donc $k = 26$ et le rendement $R = \frac{26}{31} \approx 0,838$.

Pour le BCH correcteur de $t = 2$ erreurs, il faut $2t = 4$ zéros consécutifs. Prenons $\alpha, \alpha^2, \alpha^3, \alpha^4$ (on connaît déjà les polynômes annulateurs de tous les α sauf celui de α^3). Le polynôme annulateur de α^3 doit s'annuler sur toutes les puissances puissance de 2 de α^3 à savoir $\alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{48} = \alpha^{17}$. Le polynôme annulateur de α^3 s'écrit donc $\mu_{\alpha^3}(x) = (x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^{24})(x+\alpha^{17})$. On sait que :

$$\alpha^6 = \alpha + \alpha^3$$

$$\alpha^1 2 = \alpha + \alpha^2 + \alpha^3$$

$$\alpha^{17} = \alpha(\alpha^8)^2 = 1 + \alpha + \alpha^4$$

$$\alpha^{24} = (\alpha^{12})^2 = \alpha + \alpha^2 + \alpha^3 + \alpha^4$$

On calcule alors $\mu_{\alpha^3}(x) = 1 + x^2 + x^3 + x^4 + x^5$
On trouve alors $g(x) = \mu_{\alpha^3}(x)\mu_\alpha(x) = 1 + x^3 + x^5 + x^6 + x^8 + x^9 + x^{10}$

Remarque : Les calculs n'ont pas été développés dans ce document pour la lisibilité ainsi que la complexité de les écrire avec Latex

On trouve alors un degré de 10 soit $k = 21$ puis un rendement $R = \frac{21}{31} \approx 0,677$

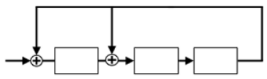
1.2 Implémenter une opération de codage standard (systématique) en utilisant la méthode de codage fournie en annexe

On utilise la technique des registres, illustrée dans l'exemple ci-dessous :

Le degré du polynôme générateur correspond au nombre de registres et les puissances des différents éléments donnent les positions respectives des différents sommateurs binaires.

Pour le correcteur d'une unique erreur on a g de degré 5, donc 5 registres et les différentes puissances (autres que celle du degré maximal, qui correspond au retour) sont 0 et 2. Il y a de fait deux sommateurs pour les registres associés.

To illustrate the encoder, we take the example of the BCH code $(7,4,3)$ where the generator polynomial is $g(x) = 1+x+x^3$. The encoder calculates $m(x) \bmod g(x)$. Let $m(x) = b_0 x + b_1 x^2 + b_2 x^3$



- Initially, the shift register contains all zeros.
- The coefficients of m(x) are clocked into the shift register one bit coefficient at a time, beginning by the highest coefficient, b_2, followed by b_1 and so on.

Figure 1: Exemple d'un décodeur BCH

Pour le correcteur de 2 erreurs, il y a donc 10 registres et 6 sommateurs, aux positions 0, 3, 5, 6, 8 et 9.

Pour la forme systématique, on fait entrer dans le programme $[0, ..., 0, m]$ qui est de longueur $N_c = 31$, après complétion du mot de code m avec des 0. Le passage par les registres permet de calculer m_c puis on renvoie $[m_c, m]$ aussi de longueur 31.

Les fonctions Matlab correspondantes sont "bch_simu_1err" pour le BCH avec une erreur et "bch_simu_2err" pour celui avec 2 erreurs.

1.3 Implémenter un décodeur ML optimal pour les deux codes

Pour cela, nous allons construire une matrice de syndrome et comparer cette matrice avec le message que nous avons, nous allons suivre l'indication de la figure 2

Pour le cas 1 erreur il va juste falloir construire les e_i contenant $N_c = 31$ bits à 0 et un bit à 1 à la position i. On va ensuite passer le e_i dans les registres précédents pour faire l'opération du $modulo(g)$. On va ensuite comparer le résultat avec le mot de code. Si il y a correspondance avec un des e_i on sait alors qu'il y a une erreur à la position i et on doit alors changer changer le bit à la position i.

Pour le cas 2 erreurs il y a les e_i comme précédemment ainsi que les e_{ij} où les bits de position i et j valent 1. La encore on va comparer et déclarer une erreur à la position i en cas de correspondance avec e_i ou 2 positions en i et j si correspondance avec un e_{ij}.

Les codes correspondants sont respectivement "correcteur_1err" et "correcteur_2err" pour le correcteur 1 et 2 erreurs (attention, on construit les matrices en dehors du code pour des questions de performance).

Pour le décodage, on peut implémenter différentes méthodes, mais la méthode par syndromes semble plus facile.
Lors du décodage on a $c' = c + e$ avec e une erreur causée par le canal. On fait $(c' \mod g) = (c + e \mod g) = 0 + (e \mod g)$
On liste toutes les erreurs e_i qui peuvent être corrigées par le code, et on met dans un tableau leur modulo. Par exemple pour le code correcteur qui corrige une erreur, on crée une matrice S_1.

$$S_1 = \begin{bmatrix} e1 \mod g \\ e2 \mod g \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ e31 \mod g \end{bmatrix}$$

Pour cet exemple $e_i = [0...010...0]$ avec le 1 à la ième position. Pour le décodage, on compare $(c' \mod g)$ avec chacune des lignes de S_1 et on en déduit l'erreur e_i. Pour le deuxième code correcteur, la matrice sera beaucoup plus grande.
Une fois l'erreur corrigée, on peut aisément récupérer le message m car $c = [m_c m]$, d'où l'utilité de la forme systématique.

Figure 2: Example of a BCH decoder

1.4 Tracer la courbe BER en fonction de $\frac{E_b}{N_0}$ pour le canal BPSK non-codé ainsi que pour les deux codes BCH (modulés en BPSK) avec un canal AWGN. $\frac{E_b}{N_0}$ peut être compris entre 0 et 10dB. Comparer les gains de codage empiriques avec la théorie.

Dans le cas non codé, la matrice H vaut la matrice identité. Le mot de code m va passer dans le BCH pour donner c. Puis c va passer dans la BPSK pour donner s. On ajout alors w, bruit blanc gaussien, à s pour simuler le canal. Puis à l'aide d'un détecteur à code on va trouver s', puis on trouve c' et le décodeur donnera m'. Il suffit alors de calculer $\frac{sum|m'-m|}{K_c}$ pour trouver le BER. On moyennera sur un grand nombre d'essai (de l'ordre de 10^5 pour avoir un BER convenable).

Pour cela on utilisera les codes fournis. Le calcul du BER théorique se fera avec la fonction "BER_th".

Les courbes théoriques et expérimentales dans le cas sans codage se superposent parfaitement, les calculs semblent donc bon.

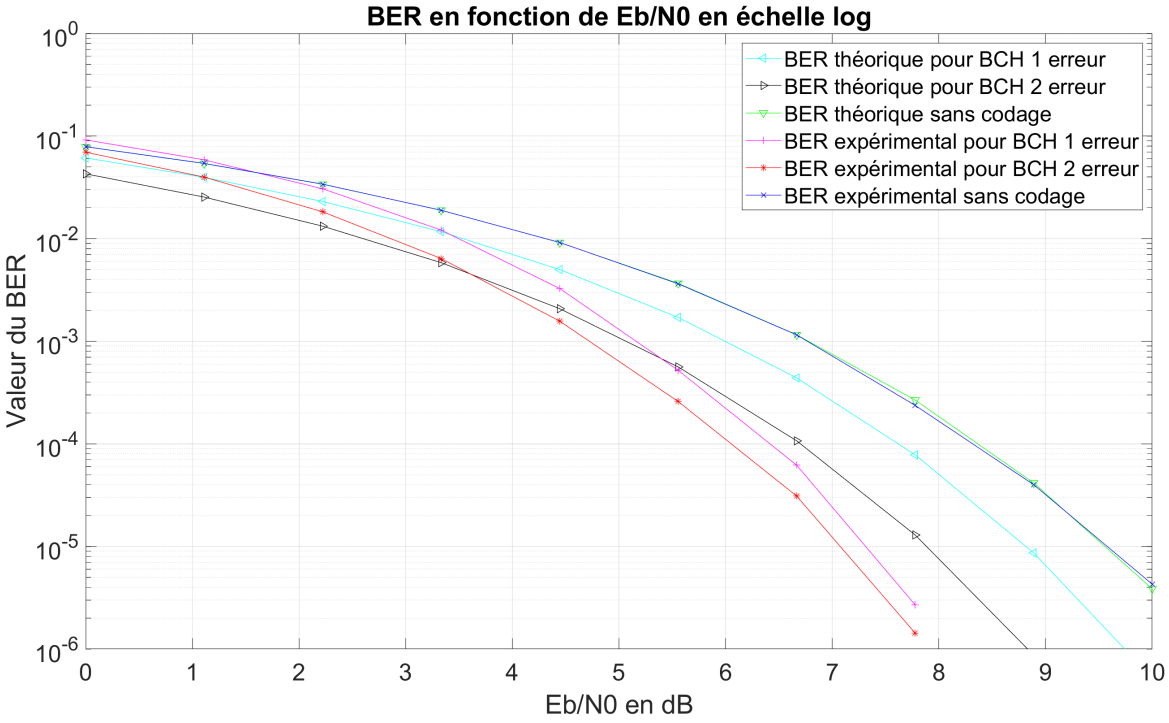


Figure 3: Tracé BER expérimental avec codage

On remarque bien que les codes correcteurs d'erreurs permettent de gagner en BER, ce qui est bien voulu.

1.5 Tracer h pour les canaux 1, 2 et 3, avec $T_s = 0.05\mu s$

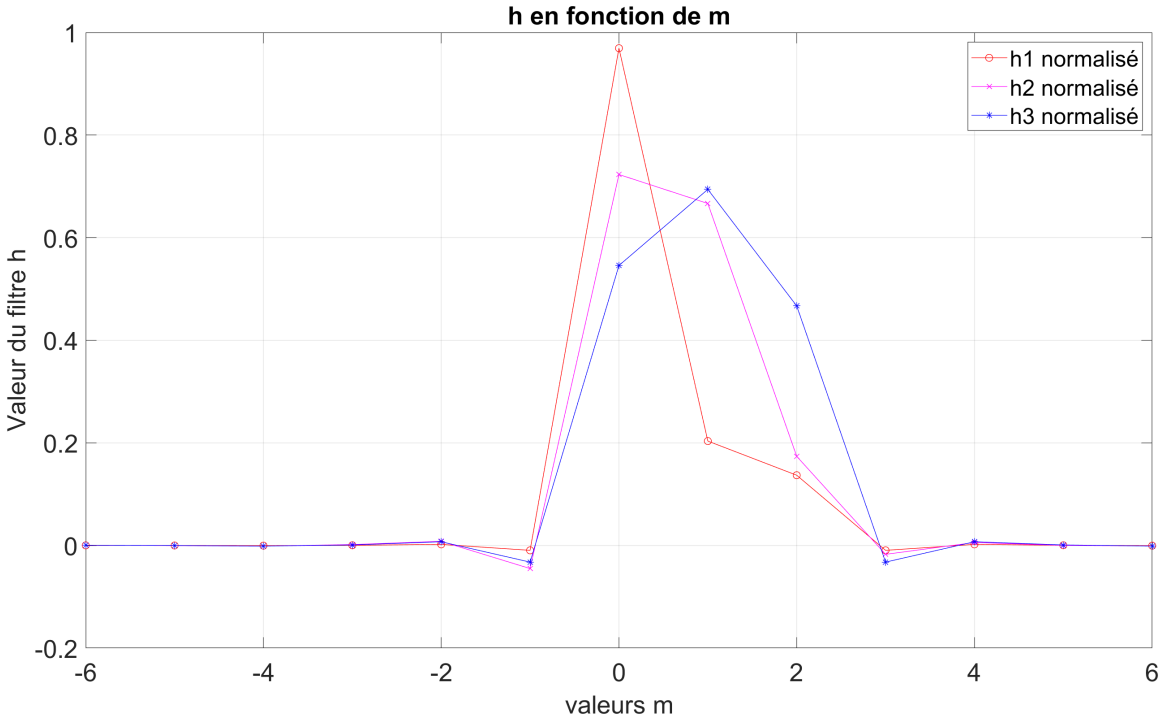


Figure 4: Plot of the h for each channel

Les figures sont beaucoup plus larges pour h2 et surtout h3 : l'influence des symboles voisins est bien plus élevée que dans le canal 1 par exemple (où h1 est presque un pic)

Il y a beaucoup plus d'interférences pour les canaux 2 et 3 ce qui donnera donc des résultats moins bons en terme de performance (ce que nous verrons dans la suite).

Le code se trouve dans le script 'plot_channel' du dépôt Git.

1.6 Tracer le BER en fonction de $\frac{E_b}{N_0}$ pour chaque canal avec trois différents equalizers (threshold, ZF et DFE) avec une modulation BPSK et sans codage de canal. On considérera typiquement $\frac{E_b}{N_0}$ compris entre 0 et 20dB.

Pour le ZF on prend P tel que PH $= I_d$ où H est la matrice de Toepliz formée par le h du canal et tel que $z = hs + w$. On a alors $z' = s + w'$ où $w' = Pw$.
Pour le DFE, on décompose H selon la décomposition QR soit $H = VT$ avec V unitaire et T triangulaire supérieure. Donc $z = VTs + w$ et :

$$z' = V^H z$$

$$z' = Ts + V^H w$$

$$z' = Ts + w'$$

Dans la suite, on plot sur une même courbe les equalizer DFE, ZF et Threshold ainsi que le BER théorique calculé selon :

$$P_e = N_{min} Q\left(\sqrt{\gamma \frac{E_b}{N_0}}\right)$$

où γ représente le gain de la modulation. N_{min} et γ dépendent de la modulation.

On trace les fonctions à l'aide de 'BER_plot' (seul les premiers arguments sont à changer).

Les différents figures sont disponibles en fichier .fig (modifiables avec Matlab) sur le dépôt Git (dans d1/fi et les images png sont dans d1/png).

Par la suite, les conventions suivantes sont établies : en rose le Threshold, en bleu le ZF et en rouge le DFE

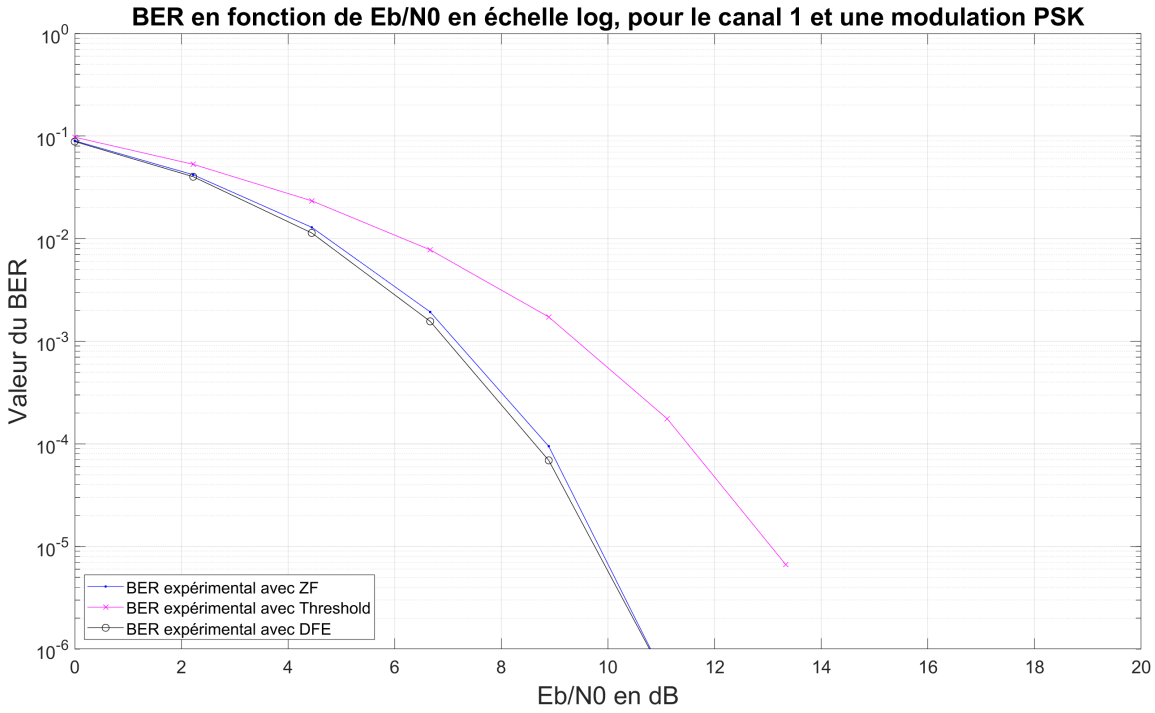


Figure 5: BER pour le canal 1 pour la BPSK

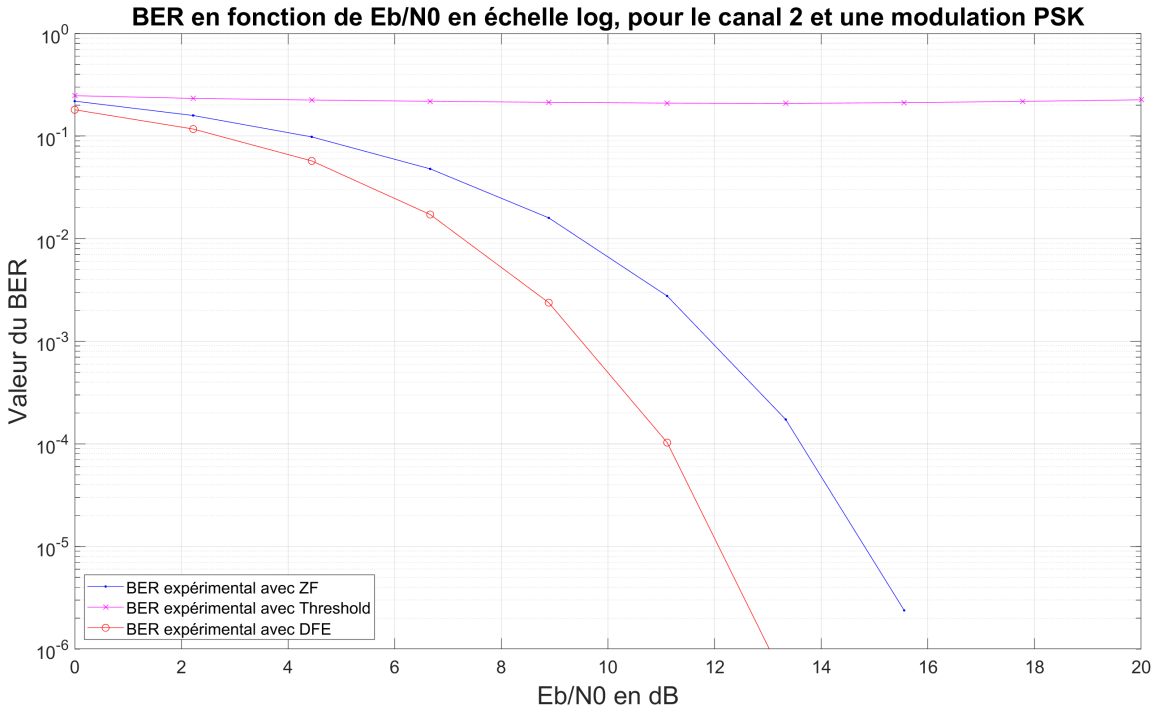


Figure 6: BER pour le canal 2 pour la BPSK

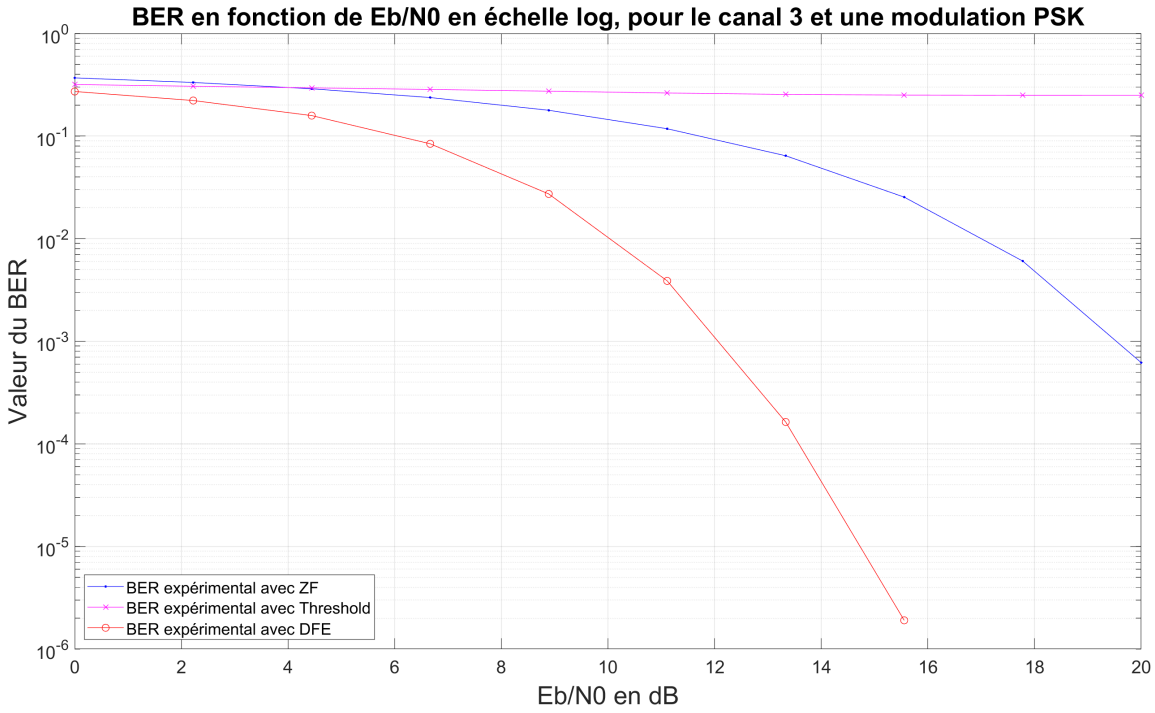


Figure 7: BER pour le canal 3 pour la BPSK

Comme attendu, le threshold donne les plus mauvais résultats. On retrouve également le résultat attendu : le DFE donne de meilleurs résultats que le ZF, surtout pour les canaux 2 et 3 qui ont beaucoup d'interférences. Le canal 1 n'a pas tellement d'interférences et ces deux méthodes donnent de bon résultats.

On remarque aussi une grande dégradation des performances (le canal 3 est pire que le 2 qui est pire que le 1, ce que le plot des h laisser suggérer)

1.7 Idem avec une modulation 8-QAM et $\frac{E_b}{N_0}$ compris entre 0 et 30dB

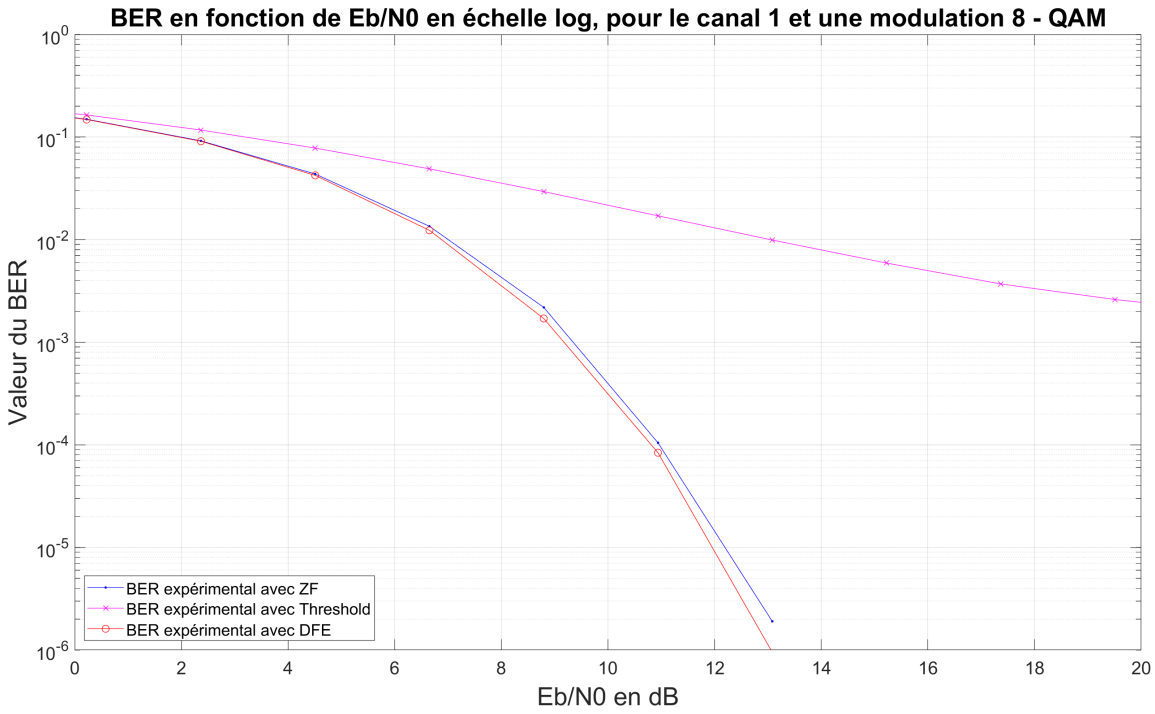


Figure 8: BER pour le canal 1 pour la 8QAM

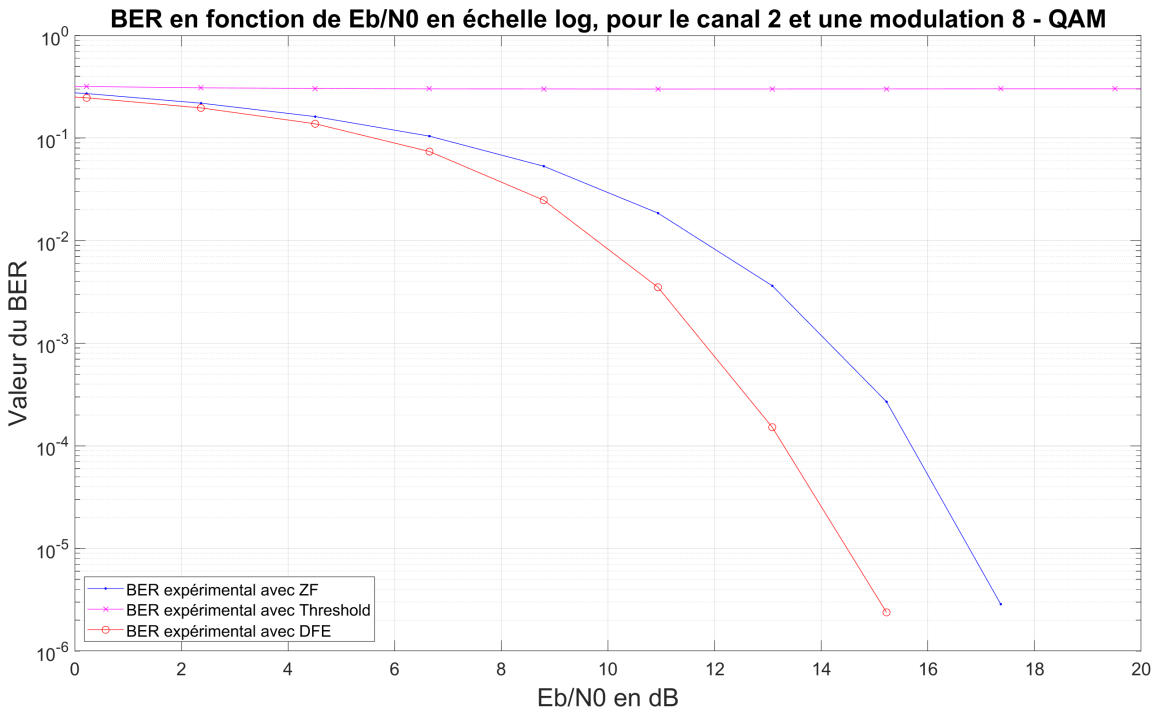


Figure 9: BER pour le canal 2 pour la 8QAM

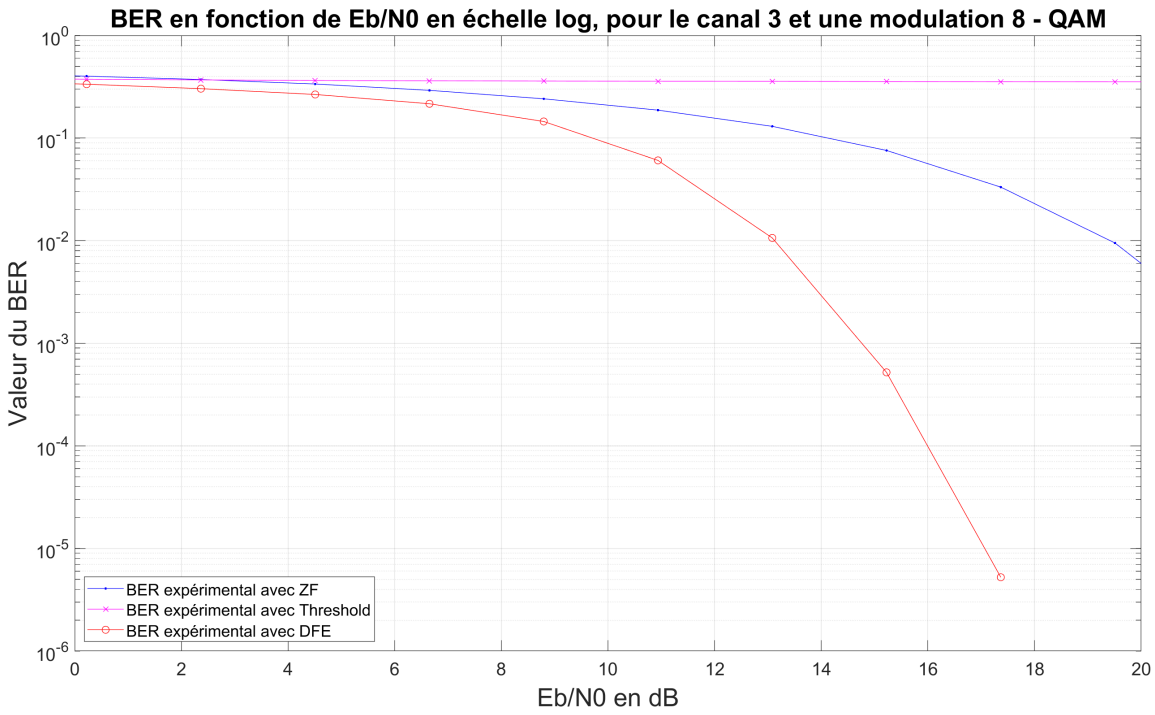


Figure 10: BER pour le canal 3 pour la 8QAM

La encore les remarques sont identiques avec la BPSK. On remarque également que pour un SNR donné, le BER est plus élevé avec la QAM qu'avec le BPSK, on retrouve donc les résultats attendus.

1.8 Idem avec une modulation 16-QAM et $\frac{E_b}{N_0}$ compris entre 0 et 30dB

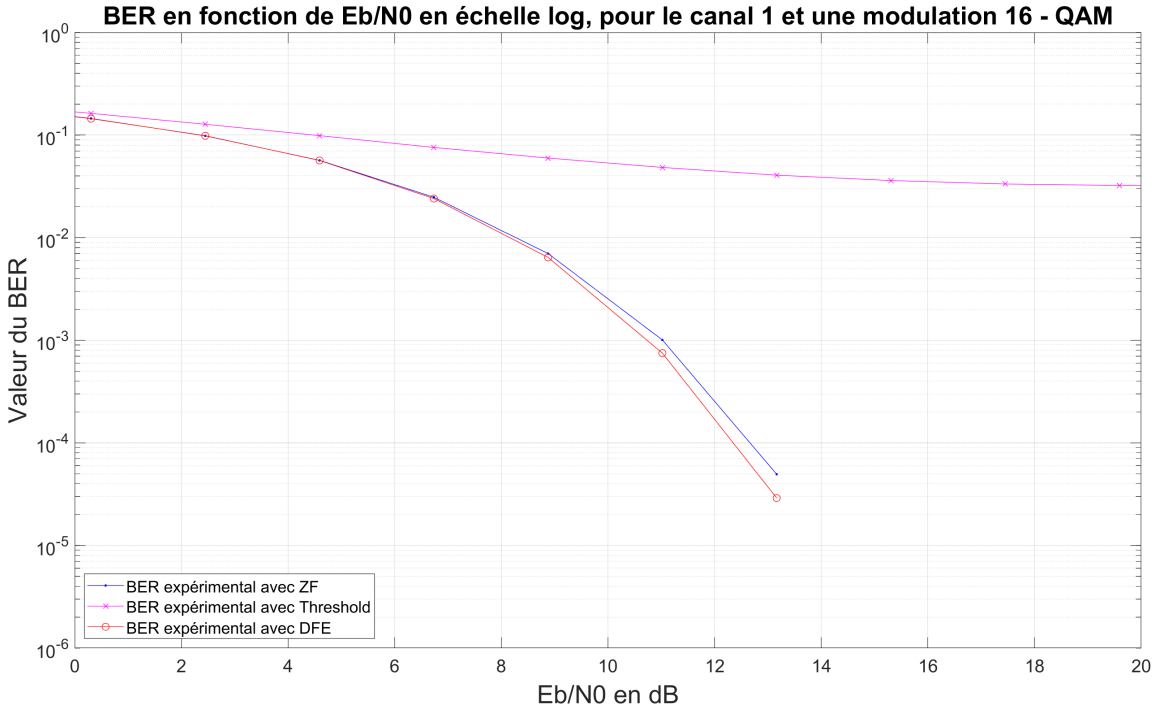


Figure 11: BER pour le canal 1 pour la 16QAM

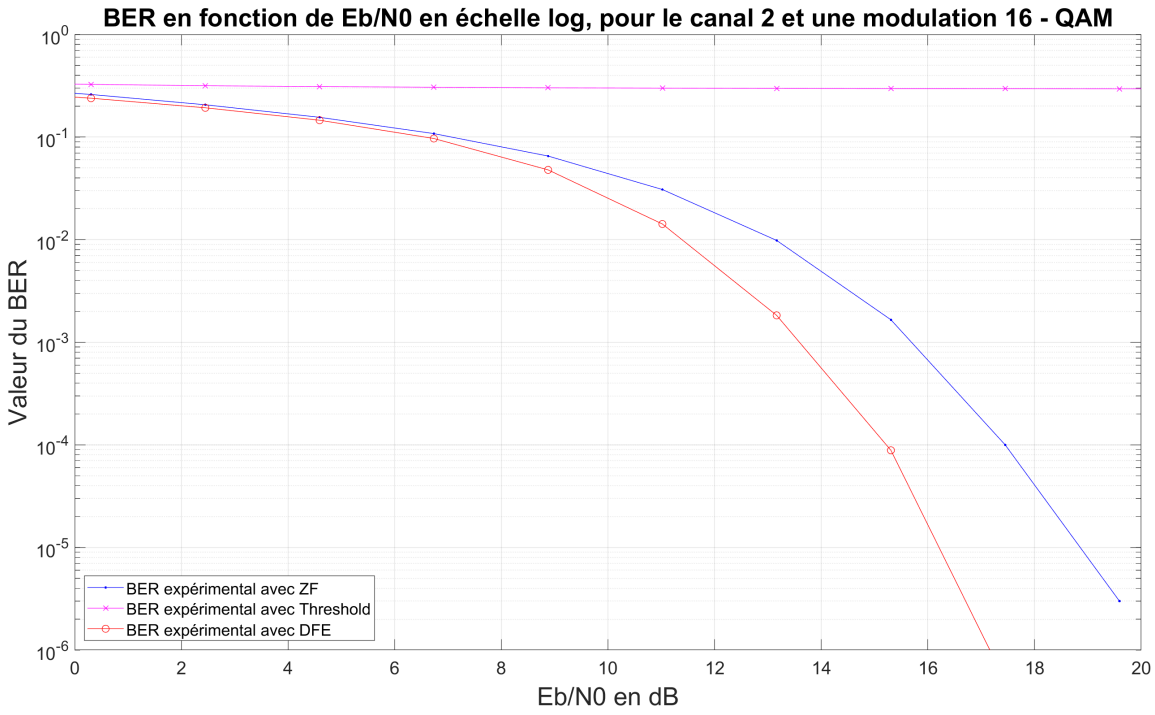


Figure 12: BER pour le canal 2 pour la 16QAM

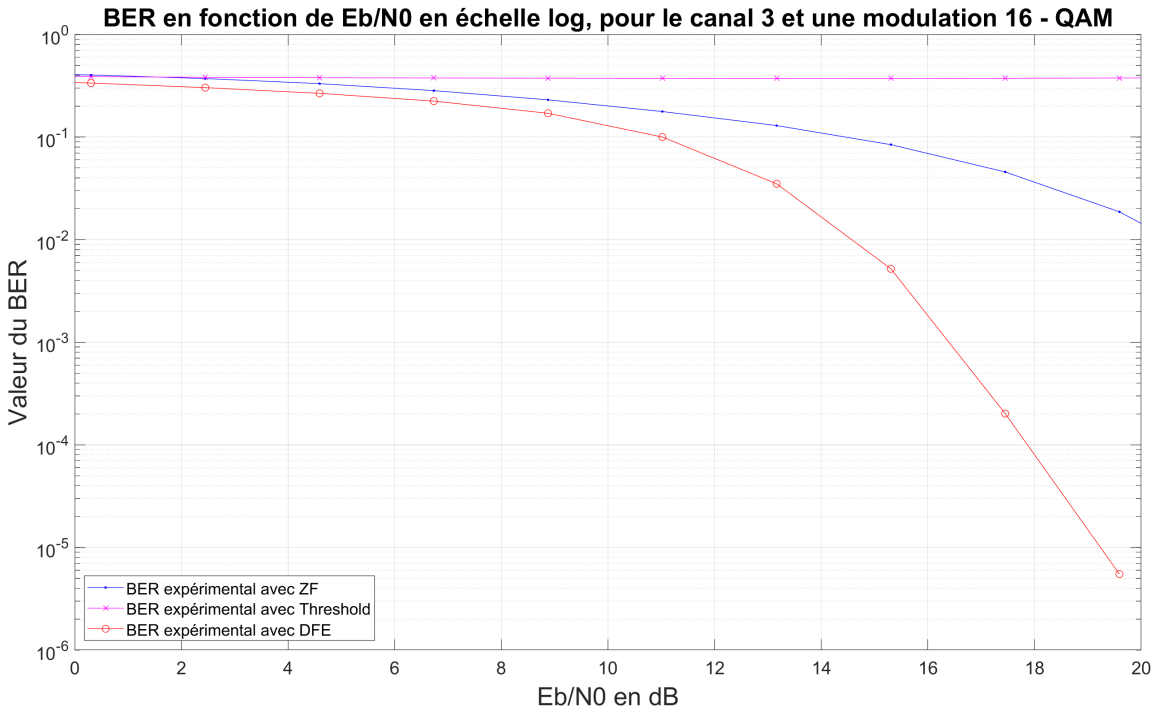


Figure 13: BER pour le canal 3 pour la 16QAM

Ici, les conclusions sont encore une fois les mêmes et pour un SNR donné le BER de la 16 QAM est plus élevé que pour la 8 QAM, ce qui est attendu.

1.9 Conclusion

Ainsi se conclut la première partie de ce projet TELECOM205. Nous avons utilisé Matlab pour déterminer les conditions de modulation et de détection optimales, et ce pour différents types de canaux. Nous nous sommes familiarisés avec l'environnement de développement et la programmation collaborative utilisant Git, ce qui nous servira pour la suite du projet avec notamment la simulation de la chaîne d'émission/réception hardware.

2 Extra-credit : code correcteur

L'extra-credit traité dans cette partie est la mise en place d'un code correcteur pour les trois canaux. Le code correcteur est le code traité précédemment (BCH

1 ou 2 erreurs) et est appliqué pour le DFE, Threshold et ZF. Est à noter que seulement une BPSK est utilisée ici.

Les fonctions utilisées sont les fonctions *BERThresCanalErr*, *BERZFErr* et *BERDFEErr*. Les courbes sont au format .fig dans le dépôt Git, dans le fichier *fig* et portent le nom *ExtraCreditChannelX* où X vaut 1, 2 ou 3 en fonction du canal désiré.

Voici les courbes obtenues où le code suivant a été utilisé : en rose le cas Threshold, en rouge le ZF, en bleu le DFE, avec des "o" le cas 1 erreur de corrigé, avec "x" le cas 2 erreurs et avec "^" pas de codage (sans BCH).

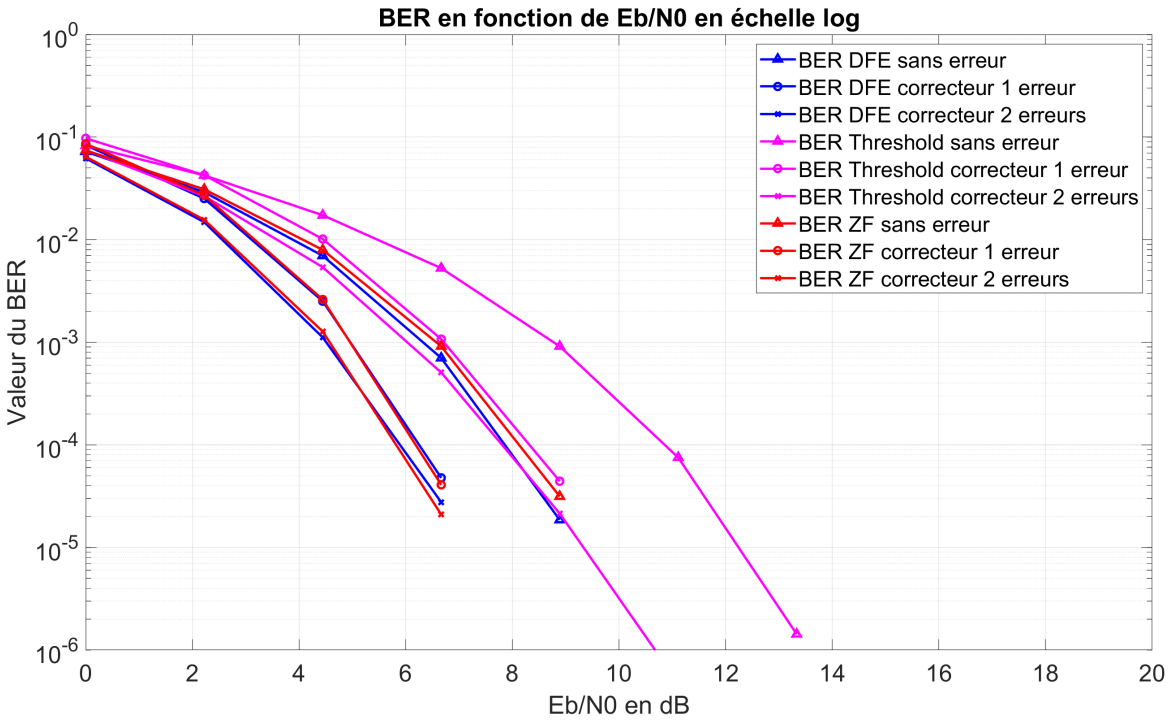


Figure 14: BER pour le canal 1 en fonction de Eb/N0

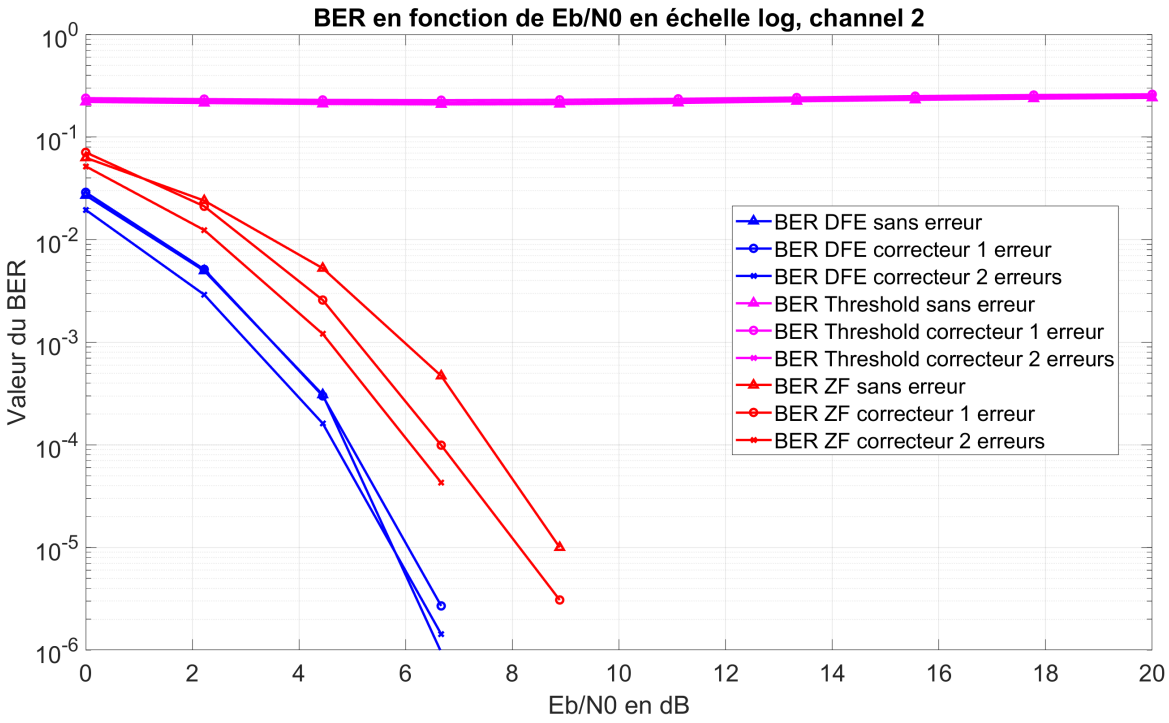


Figure 15: BER pour le canal 2 en fonction de Eb/N0

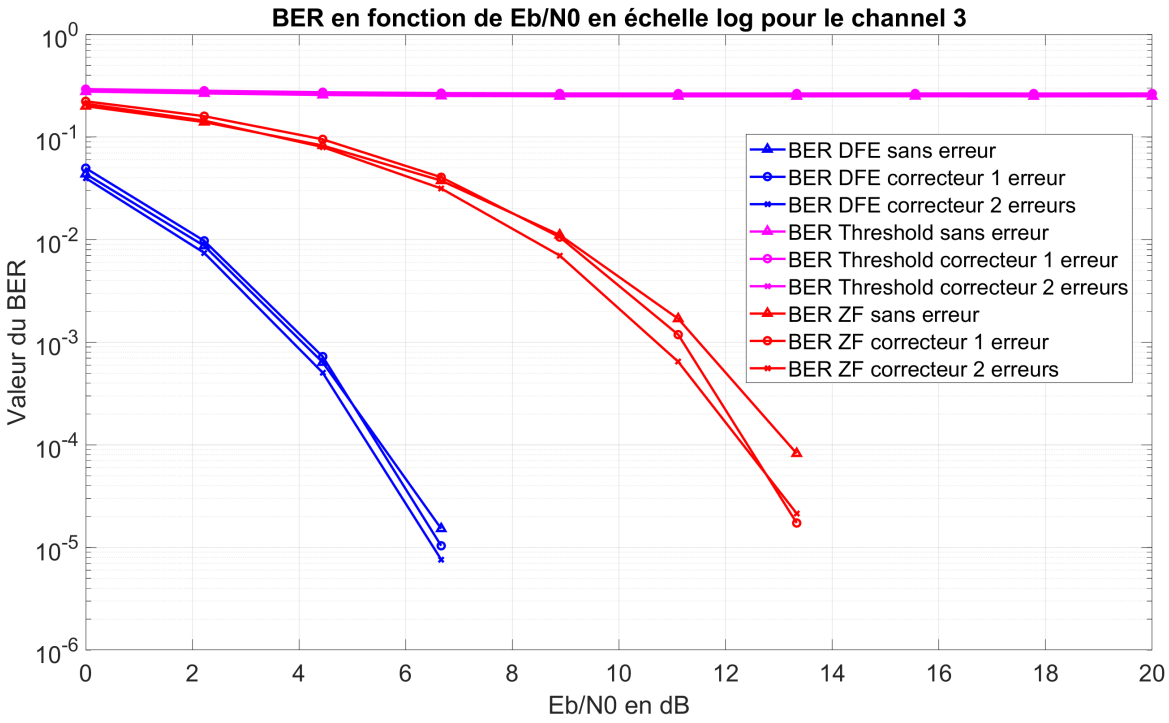


Figure 16: BER pour le canal 3 en fonction de Eb/N0

Comme attendu et comme observé dans les premiers cas, le canal 1 est bien meilleur que le 2 et le 3.

De plus, comme on pourrait s'y attendre, le codage permet d'atteindre des BER encore plus petit, d'autant meilleur qu'on corrige un plus grand nombre d'erreur.

En regardant DFE, threshold ou ZF on aperçoit que le DFE donne toujours les meilleurs résultats, indépendamment du fait qu'il y ait un codage ou non. Au contraire, le threshold donne les pires résultats.

Pour le threshold et les canaux 2 et 3, au vu de leur performance médiocre, le code correcteur n'apporte rien. Cependant, on a un faible gain pour DFE et ZF, gain plus important pour ZF que DFE cependant.

Enfin, pour le canal 1, les 3 méthodes sont plus proches en terme de résultat mais les correcteurs restent toujours les meilleurs en terme de performance. Est à noter que que le correcteur 2 erreurs du threshold est presque aussi bon que le DFE sans correcteur alors que dans tous les autres cas il est bien pire.

3 Extra-credit : ARQ

Dans cette partie, on implémente une protocole ARQ avec au plus une retransmission. Si une trame envoyée n'est pas effectivement décodée à la réception on la renvoi. On s'intéresse donc au 'throughput', c'est à dire le rapport entre le nombre de bits de trames effectivement décodées et entre le nombre de symboles transmis.

On a donc 3 cas :

- Si rien n'est bien transmis (il y a au moins un symbole différent entre reçu et émis), même après l'ARQ on renvoit 0.
- Si le message est bien transmis du premier coup : on aura par exemple dans le cas d'une 16 QAM de longueur 128 : $\frac{128}{\frac{128}{log_2(16)}}$. Avec le log_2 pour la taille des symboles dans la modulation utilisée.
- Si le message est bien transmis du second coup, on va avoir pareil que précédemment avec un facteur 2 au dénominateur (car on aura retransmis le message).

Pour les différentes configurations (BPSK, 8 ou 16 QAM, avec ou sans codage BCH) on va utiliser des trames de petites taille (31 pour BPSK, 3*31 pour 8 QAM et 4*31 pour 16 QAM, où 3 et 4 sont utilisés pour avoir des divisions entières avec les tailles des modulations). L'usage de petite taille va permettre d'avoir des résultats plus concluant (plus rapide que des grandes tailles de trame mais également plus précis : en augmentant la taille des trames on augmente le taux d'erreur -en probabilité on a plus d'apparition d'erreurs si on a plus de bits-).

On va ensuite moyenner sur une centaine de courbe pour avoir les bonnes allures de courbe. On trace le throughput en fonction de $\frac{E_s}{N0}$ où $E_s = 1$ pour la BPSK (on prend des symboles d'amplitude 1), et $\frac{2(C-1}{3}$ pour une QAM, avec C la taille de la constellation (8 pour une 8 QAM et 16 pour une 16 QAM). Le bruit (N0) sera pris constant pour les 3 modulations, il varie entre 0 et 30dB

On obtient la courbe suivante :

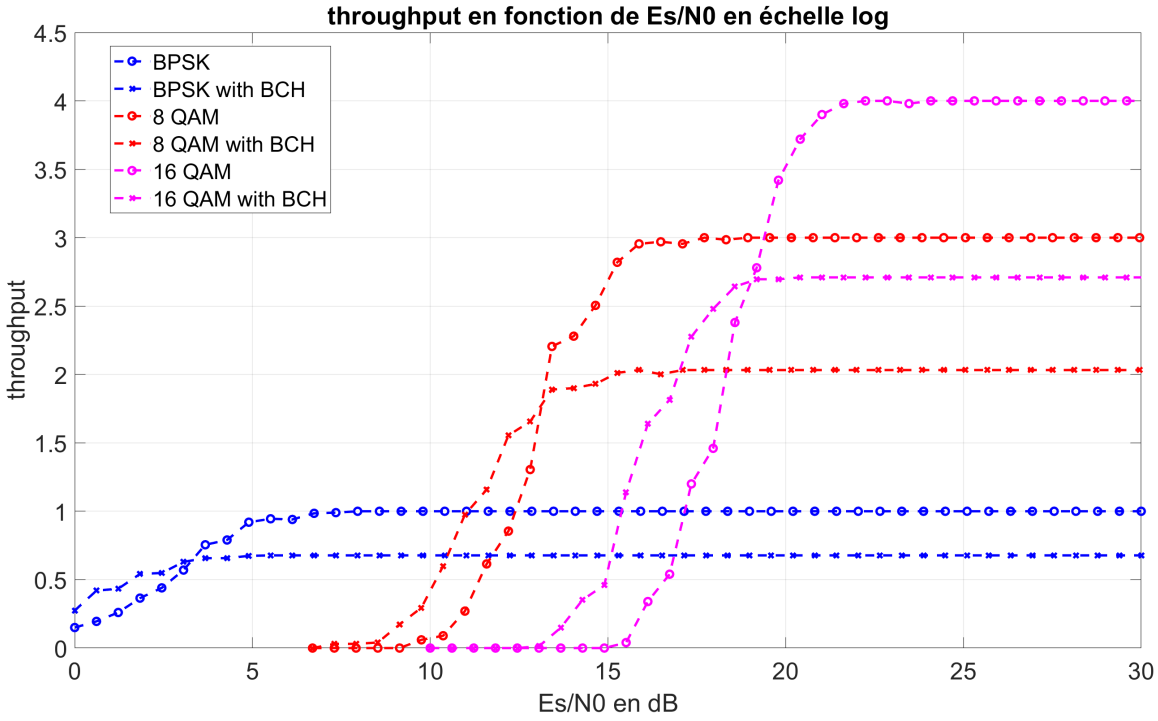


Figure 17: Throughput pour différentes constellations en fonction de Es/N0

On remarque que pour des SNR élevés on atteint un seuil pour les différentes configuration. Ce seuil peut être retrouvé assez facilement.

- Pour une BPSK seule, au maximum on a un throughput de $\frac{31}{31} = 1$. Pour rappel les trames en BPSK sont de longueur 31.
- Pour une 8 QAM seule, on a cette fois $\frac{3\times31}{\frac{3\times31}{log_8(8)}} = 3$. Pour rappel les trames en 8 QAM sont de longueur 3x31.
- Pour une 16 QAM seule, on a cette fois $\frac{4\times31}{\frac{4\times31}{log_8(16)}} = 4$. Pour rappel les trames en 8 QAM sont de longueur 4.31.
- Pour une BPSK avec codage, on a cette fois un gain de codage de $\frac{21}{31} = 0.677$ donc un throughput de (on multiplie le throughput précédent par le gain) de 0.677, ce qui est bien sur la courbe.
- Pour une 8 QAM avec codage, on a cette fois un gain de codage de $\frac{3\times21}{3\times31} = 0.677$ donc un throughput de (on multiplie le throughput précédent par le gain) de 2.03, ce qui est bien sur la courbe.

- Pour une 16 QAM avec codage, on a cette fois un gain de codage de $\frac{4\times21}{4\times31} = 0.677$ donc un throughput de (on multiplie le throughput précédent par le gain) de 2.7, ce qui est bien sur la courbe.

Comme convenu, plus le SNR est élevé plus le trhoughput est important. Grâce au gain des symboles, la 16 QAM est plus performante que la 8 QAM, elle même plus performante que la BPSK.

Avec codage on retrouve bien un throughput plus faible (courbes toujours en dessous sur la figure). En effet on l'a exprimé de manière théorique juste au dessus mais plus précisément : le codage rajoute des bits de redondance donc augmente le dénominateur comparativement au numérateur. On a un débit plus faible mais moins d'erreurs en général. C'est d'ailleurs ce qu'on aperçoit : le throughput est non nul pour des $\frac{E_s}{N0}$ plus faible pour le cas sans codage, à constellation identique. C'est également ce qu'on voyait dans la partie précédente : les BER avec codage sont plus faible. Ainsi pour des rapports signal-à-bruit $\frac{E_s}{N_0}$ faibles on privilégiera les cas avec codage mais si on veut vraiment un throughput le plus élevé possible au détriment du BER on prendra pour des hauts $\frac{E_s}{N_0}$ le cas sans codage.