

# Signal - TP3 - Thème2

Maéva Bachelard - Margot Laleu

# Fonctions MatLab

**rand(x)** : renvoie 1 nombre aléatoire suivant la loi uniformément répartie sur  $[0, 1]$ .

**rand(x, y)** : renvoie une matrice de format  $x*y$  contenant des nombres aléatoires suivant la loi uniformément répartie sur  $[0, 1]$ .

**randn(x)** et **randn(x,y)** : fonctionnent comme rand sauf que les nombres aléatoires qu'elles renvoient suivent une loi normale centrée en 0 et d'écart type 1.

**histogram(x)** :

trace l'histogramme du vecteur x en choisissant automatiquement les regroupements de largeur uniforme à effectuer.

**histogram(x,n)** :

trace l'histogramme du vecteur x en faisant n regroupements

**histogram(x,e)** : (e un vecteur)

trace l'histogramme du vecteur x en faisant les regroupements sur les intervalles :

$[e(1);e(2)], [e(2);e(3)] \dots [e(N-1);e(N)]$

**stem(x)** : affiche une séquence de donnée discrète sous forme de kronecher.

**erfc(x)** : retourne la fonction complémentaire d'erreur évaluée pour chaque valeur du vecteur x.

**sign(x)** : pour x réel, renvoie un tableau de la même taille que x dont les valeurs sont :

-1 si l'élément correspondant de x est  $<0$ ,  
+1 s'il est  $>0$   
0 s'il est égal à 0.

# I Canal idéal de Nyquist AWGN

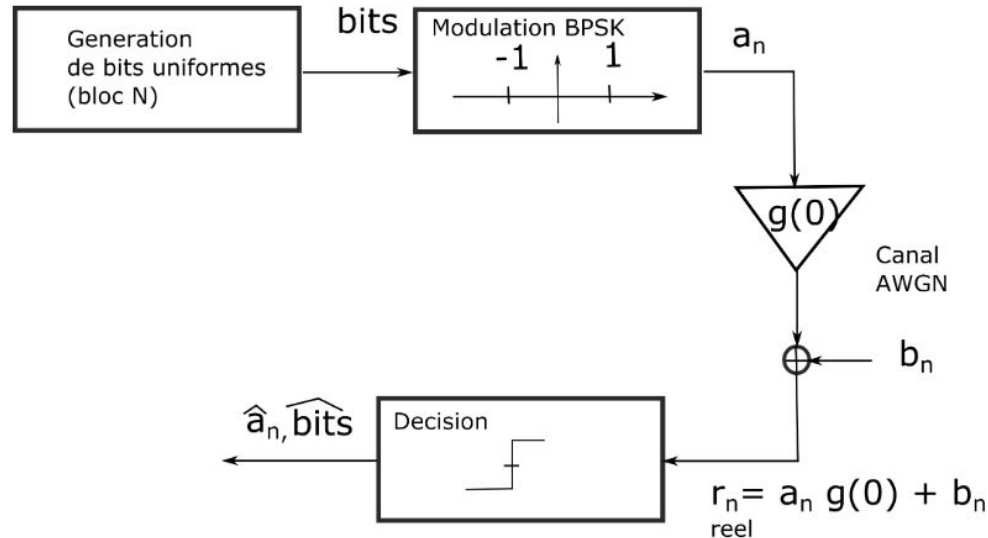
Notre premier objectif de ce TP est de simuler un canal de Nyquist idéal, dont le bruit est Gaussien, afin d'en observer la probabilité d'erreur binaire à la réception.

Pour cela nous créons tout d'abord **3 fonctions** :

**genBPSK** qui renvoie un signal de type BPSK dans un vecteur de taille N. La répartition entre les -1 et les 1 suit une loi équi-probable.

**genBruitN** renvoie un vecteur de taille N, dont les valeurs suivent une loi normale (= Gaussienne) de moyenne nulle et d'écart type  $\sigma_b$ .

**decodeBPSK** qui renvoie un vecteur de taille N correspondant au signal de sortie du canal. Il renvoie le signe du signal réellement reçu  $r_n$ .

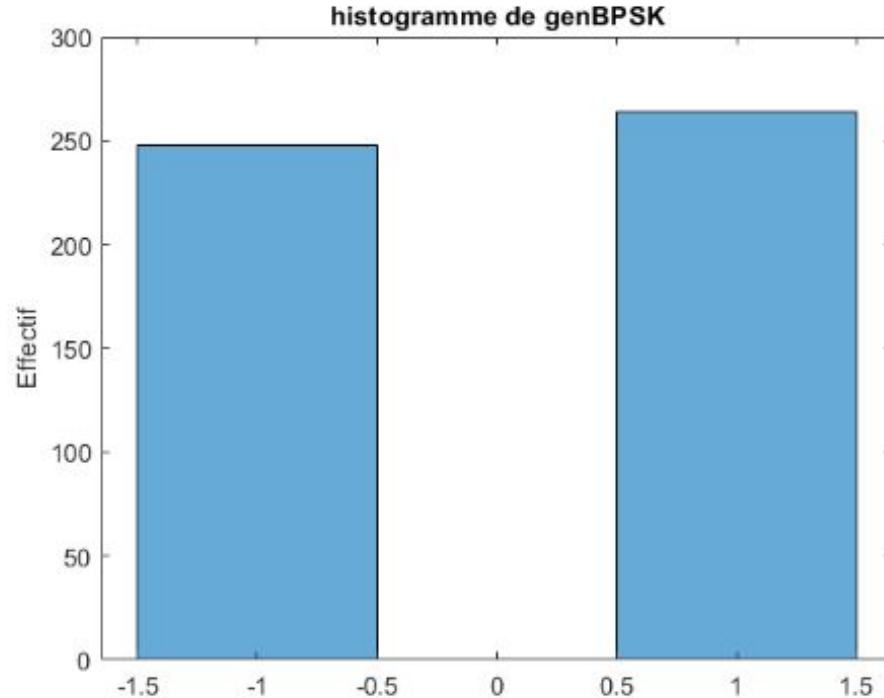


# Fonction genBPSK

Codage de genBPSK : on crée tout d'abord un vecteur de taille N rempli suivant la loi uniforme sur [0, 1] grâce à la fonction rand. Ensuite on soustrait 0.5 à chaque valeur, puis on prend le signe de la soustraction. Comme les termes d'origine suivent une loi uniforme il est supposé y avoir autant de valeurs négatives (et donc transformée en -1 par la fonction sign) que de valeurs positives (transformée en +1). On devrait donc observer environ 50% des termes à -1 et 50% des termes à +1, et aucun à 0 par nullité, pour une loi continue, de la probabilité de tomber sur une valeur précise.

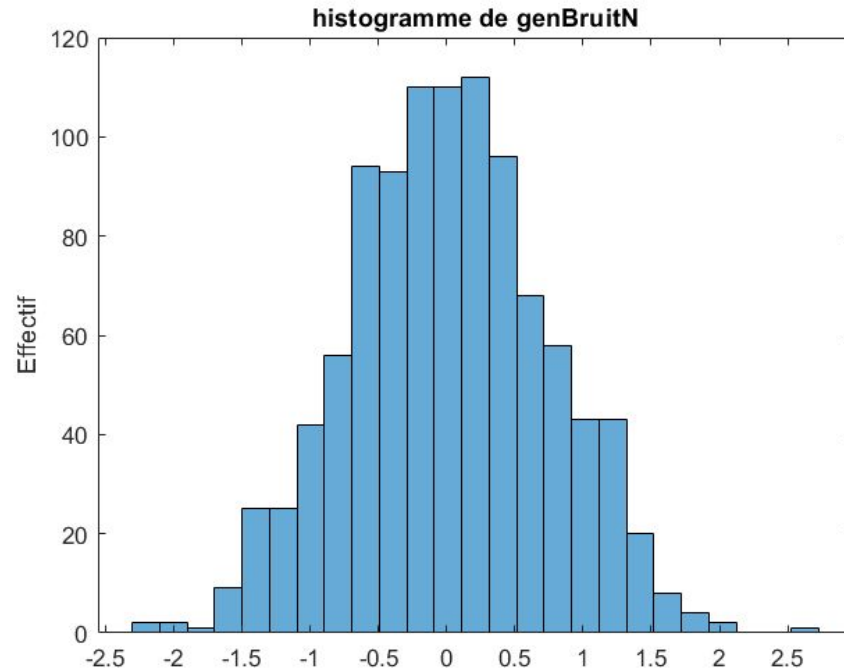
Le résultat que l'on obtient est donc cohérent.

On travaille avec  $N = 512$ ,  
 $\sigma_b^2 = 0.5$  et  $g(0) = 1$ .



# Fonction genBruitN

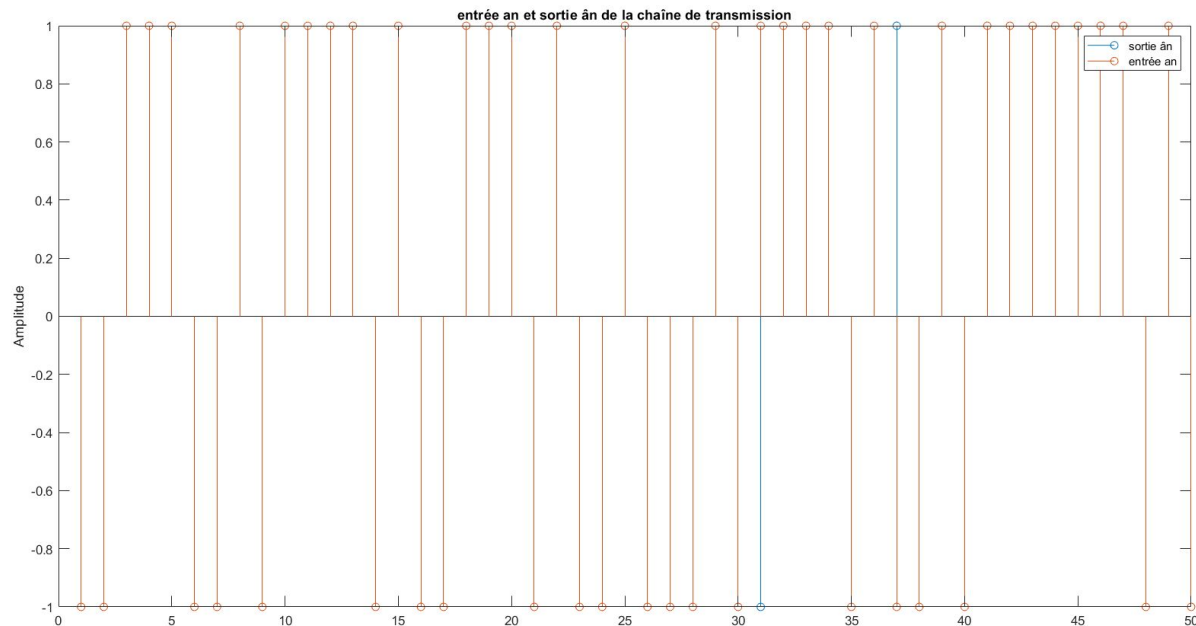
genBruitN renvoie un vecteur de taille N, dont les valeurs suivent une loi normale de moyenne nulle et d'écart type  $\sigma_b$ . Elle est codée grâce à la fonction randn multipliée par  $\sigma_b$ . Pour N = 1024 on obtient la courbe ci-dessous, ce qui correspond bien à une loi normale de moyenne nulle (centrée en zéro).



# Comparaison de $a_n$ et $\hat{a}_n$

On travaille toujours avec  $N = 512$ ,  $\sigma_b^2 = 0.5$  et  $g(0) = 1$ .

On superpose les deux courbes. S'il n'y avait pas d'erreur on ne devrait voir que l'une des deux courbes, l'autre étant cachée par la première puisque identique, cependant on remarque qu'il y a des erreurs puisque qu'on voit les échantillons (le 31 et le 37ème ici) qui diffèrent de l'entrée dans la chaîne.



Notre signal possède bien 512 échantillons, nous avons simplement limité son affichage à 50 échantillons pour une meilleure lisibilité.

# Taux d'erreur binaire à la réception

On compte le nombre d'erreurs via le code ci-dessous, répondant bien aux différents cas explicités dans le tableau. On obtient ainsi un taux d'erreur entre 7 et 8 % suivant les lancers.

```
NBerreur = sum(sign(abs(Anf-An)))  
TauxErreur = NBerreur/N
```

An	AnF	abs(Anf-An)	sign(abs(Anf-An))
1	1	0	0
1	-1	2	1
-1	1	2	1
-1	-1	0	0

```
NBerreur = 37
```

```
TauxErreur = 0.0723
```

# Variance du bruit

Le lien entre le RSB

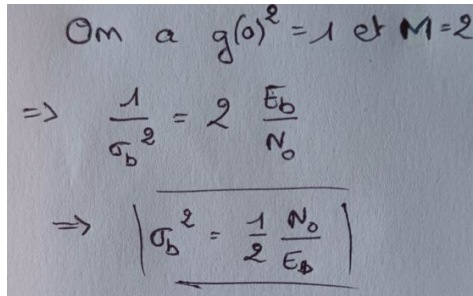
$$\text{SNR} = \frac{g(0)^2}{\sigma_b^2}$$

et le RSB normalisé **SNR normalisé**  $E_b/N_0$  est :

$$\frac{g(0)^2}{\sigma_b^2} = \frac{6 \log_2 M}{M^2 - 1} \frac{E_b}{N_0}$$

avec  $M = 2$  et  $n = \log_2 M = 1$  bit car un signal BPSK possède 2 symboles (-1 et 1).

On calcule la variance  $= \sigma_b^2$  grâce à la formule et au code MatLab ci-dessous.



On a  $g(0)^2 = 1$  et  $M=2$   
 $\Rightarrow \frac{1}{\sigma_b^2} = 2 \frac{E_b}{N_0}$   
 $\Rightarrow \sigma_b^2 = \frac{1}{2} \frac{N_0}{E_b}$

```
RSBNdB = 0:12  
RSBN = 10.^(RSBNdB*(1/10))  
  
Var = 1./(2.*RSBN)
```

Le nombre d'erreurs et le taux d'erreurs dépendent de la variance. On crée donc les vecteurs nombre et taux d'erreurs correspondant aux différentes valeurs de la variance, via la boucle ci-contre.

```
Txerreur = zeros(1,13);  
Nberreur = zeros(1,13);  
i = 1;  
  
for s2 = Var  
    Bn2 = genBruitN(N, s2);  
    Rn2 = An + Bn2;  
    Anf2 = decodeBPSK(Rn2);  
    NBerreur = sum(sign(abs(Anf2-An)));  
    TauxErreur = NBerreur/N;  
    Txerreur(i)=TauxErreur;  
    Nberreur(i)=NBerreur;  
    i=i+1;  
end
```

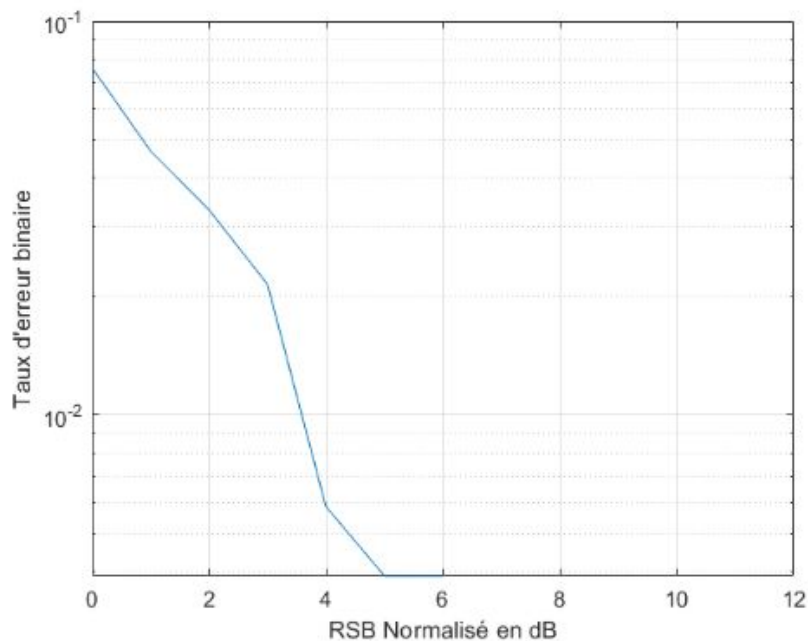


On obtient les résultats ci-dessous

RSB Normalisé dB	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>
	0	1	2	3	4	5	6	7	8	9	10	11	12
RSB Normalisé	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>
	1	1.2589	1.5849	1.9953	2.5119	3.1623	3.9811	5.0119	6.3096	7.9433	10	12.5893	15.8489
Variance	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>
	0.5000	0.3972	0.3155	0.2506	0.1991	0.1581	0.1256	0.0998	0.0792	0.0629	0.0500	0.0397	0.0315
Nombre d'erreur	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>
	38	21	18	11	10	0	0	0	0	0	0	0	0
Taux d'erreur	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>
	0.0742	0.0410	0.0352	0.0215	0.0195	0	0	0	0	0	0	0	0

On remarque ainsi que la variance diminue lorsque le RSB normalisé augmente. Et plus le RSB normalisé augmente, moins il y a d'erreurs (Pour N constant) et donc le taux d'erreur diminue.

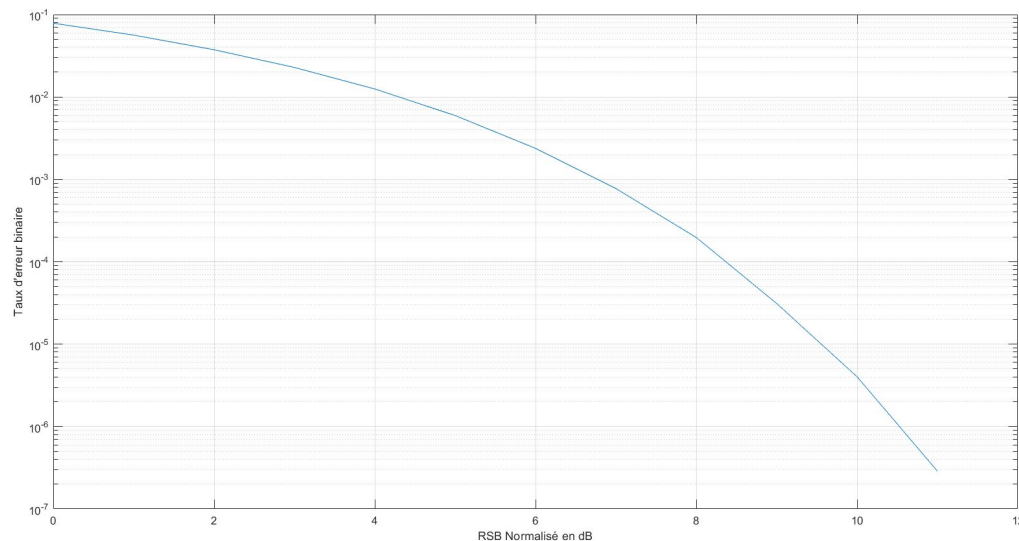
# Taux d'erreur binaire en fonction du RSB normalisé en dB



On remarque qu'à partir de RSB normalisé en dB égal à 5, le taux d'erreur est nul pour **N = 512**.

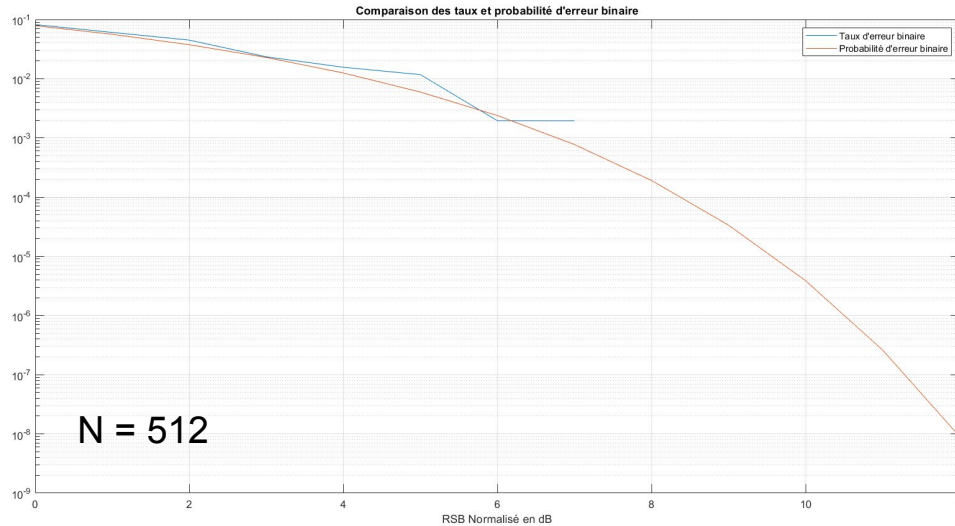
Attention à bien mettre l'axe des ordonnées en logarithmique.

Nous avons donc essayé pour **N = 10 436 608**. Cela n'est pas le N minimale pour tracer la courbe sur toute la plage [0, 12] du RSB normalisé en dB mais uniquement sur la plage [0,11] car MatLab plante pour un trop grand nombre de valeurs. Nous obtenons :



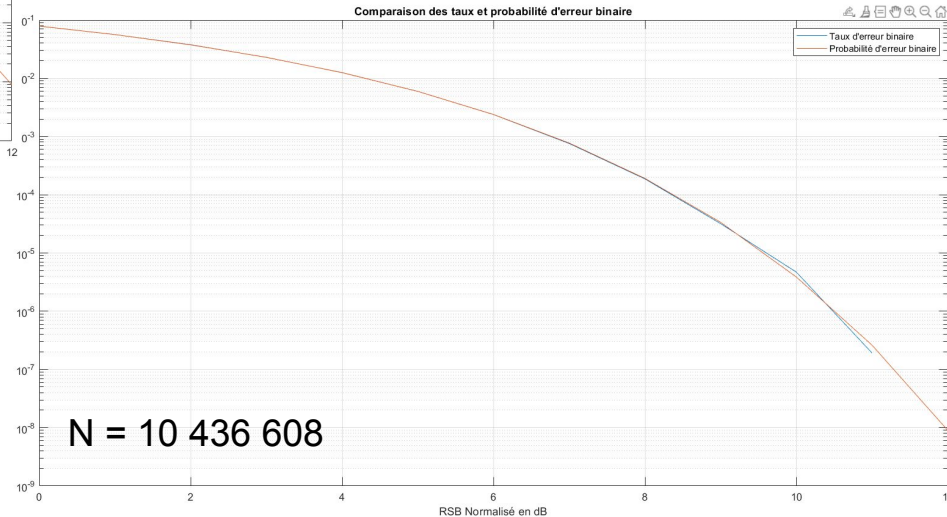
# Probabilité d'erreur théorique

Afin de comparer notre résultat à la théorie, nous allons superposer notre courbe du taux d'erreur binaire à celle de la probabilité d'erreur binaire, dont la formule pour la BPSK est la suivante :

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


Et on utilise la fonction erfc de MatLab qui est lié à la probabilité d'erreur binaire comme suit.

$$P_b = \frac{1}{2} \text{erfc} \left( \sqrt{\frac{E_b}{N_0}} \right)$$



On remarque que pour N suffisamment grand la courbe de probabilité d'erreur, donc théorique, et très proche de la courbe pratique, donc celle du taux d'erreur. Ce qui n'est pas forcément le cas pour N trop petit. (N=512 ici).

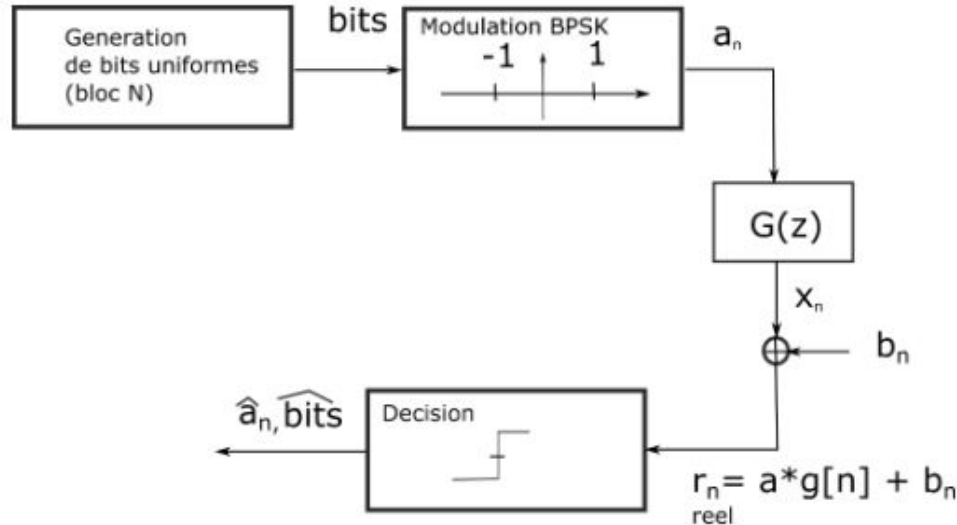
On peut ainsi lire, pour un taux d'erreur à  $10^{-3}$ , un RSB normalisé d'environ 6,4 dB.

## II Interférence entre symboles (IES) et égalisation

L'objectif est maintenant d'observer l'effet des IES dans un canal de transmission. Nous allons donc travailler sur un canal non idéal comme présenté ci-dessous.

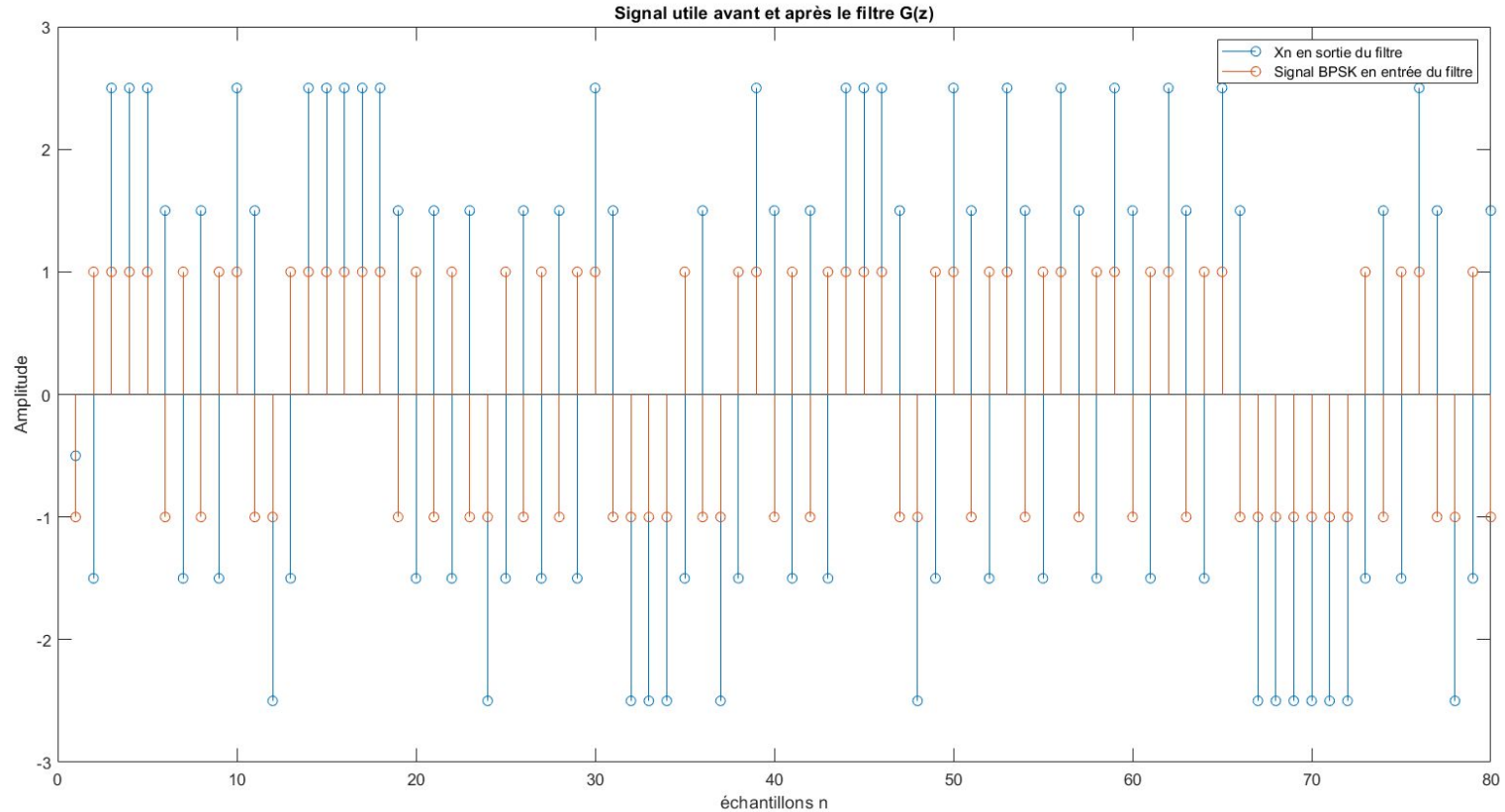
Pour simuler cela nous créons la fonction genSig qui simule le filtre  $G(z)$  appliqué à genBPSK(N).

On déduit de  $G(z)$ , l'expression de  $x[n]$ .



$$\begin{aligned} G(z) &= 0,5 + 0,2z^{-1} = \frac{y(z)}{x(z)} \\ \Rightarrow x(z)(0,5 + 0,2z^{-1}) &= y(z) \\ \Rightarrow 0,5x[m] + 0,2x[m-1] &= y[m] \\ \Rightarrow h[m] &= 0,5\delta[m] + 0,2\delta[m-1] \\ \text{Avec les notations de l'énoncé:} \\ x[m] &= (a * h)[m] = 0,5a[m] + 0,2a[m-1] \\ \text{et } a[m] &= 0 \quad \forall m < 0 \\ \Rightarrow x[0] &= 0,5a[0] \\ x[1] &= 0,5a[1] + 0,2a[0] \\ x[2] &= 0,5a[2] + 0,2a[1] \\ x[3] &= 0,5a[3] + 0,2a[2] \end{aligned}$$

# Superposition du signal BPSK et du signal après le filtre $G(z)$



On remarque alors que le filtre  $G(z)$  modifie l'amplitude des échantillons et semble inclure des erreurs binaires supplémentaires.

## Calcul du taux d'erreur binaire du canal non-idéal pour $N = 512$

En considérant qu'il y aura une erreur binaire en fin de chaîne de transmission, c'est-à-dire après ajout du bruit, si  $a[n0]$  et  $x[n0]$  sont de signes différents, soit du côté opposé de la barre de décision placée à 0 dans le cas du chaîne d codage BPSK.

### Taux d'erreur du canal non idéal

```
NBerreurBis = 0;
VectN = 1:N;

for n = VectN
    if sign(An3(n)) ~= sign(Xn(n))
        NBerreurBis = NBerreurBis +1
    end
end

TauxErreurBis = NBerreurBis/N
```

Grâce au code ci-contre, on obtient un taux d'erreur binaire avoisinant les 50%, ce qui est énorme.

```
NBerreurBis = 242
TauxErreurBis = 0.4727
```

On en déduit donc que les IES (interférence entre symboles) ont un fort impact sur la fiabilité de la transmission du signal, il sera donc nécessaire de les éviter.