TP3 : Codes correcteurs d'erreurs.

OUCHENE Souhil
Élève Ingénieur en 1ere année DSIA
École Nationale Polytechnique
Avenue des Frères Oudek, Hacen Badi, B.P.182, El-Harrach, 16200 Alger, Algérie
souhil.ouchene@g.enp.edu.dz

1. Objectif du TP :

L'objectif du TP est d'observer les améliorations apportées par les codes correcteurs d'erreur (ECCs pour Error Correcting Codes) sur les performances des transmissions numériques. Les performances sont exprimées en termes de TEB (Taux d'Erreur Binaire) et les simulations sont effectuées dans le cadre d'une transmission dans un canal BBGA (Bruit Blanc Gaussien et Additif). Quelques types de codes sont étudiés : les codes en blocs linéaires à base de matrice génératrice, les codes cycliques, les codes convolutifs. Le principe des simulations consiste à comparer les performances de deux chaînes de transmission : avec et sans ECCs. Ces performances sont évaluées en termes de TEB en fonction du rapport E_b/N_0 où E_b désigne l'énergie moyenne reçue par bit et $N_0/2$ désigne la densité spectrale de puissance du bruit additif blanc gaussien.

2. Travail a effectuer :

2.1. Code en bloc linéaire C(7,4)

2.1.1. Test sans bruit :

- **CODE MATLAB :**

```
%part1: Test sans bruit:
%generation des Nb bits:
alt=randi([0 1],100,4);
%la matrice generatrice:
G=[1 0 0 0 1 1 0; 0 1 0 0 0 1 1; 0 0 1 0 1 1 1; 0 0 0 1 1 0 1];
%codage de la series generee:
CODE = encode(alt, 7, 4, 'linear', G);
%decodage de la serie generee:
MSG = decode(CODE, 7, 4,'linear', G);
%verification:
for i = 1:400
    comparexor=0;
    comparexor(i)=xor(CODE(i), MSG(i));
end
if comparexor==zeros(1,400)
    disp('there is no error');
else
    disp('error in decoding');
end
```

- **REMARQUES :**

- On a utilisé la fonction **« randi( ) »** au lieu de la fonction **« randint( ) ».**
- On a généré une matrice de bits de taille 100*4.
- Pour la comparaison, on a utilisé la fonction **« xor( ) »** vu durant le TP.

➢ **RESULTATS :**

On peut bien vérifier que la séquence émise correspond bien à la séquence décodée, on a le texte affiché (**figure1**) :

```
there is no error
```

Figure1

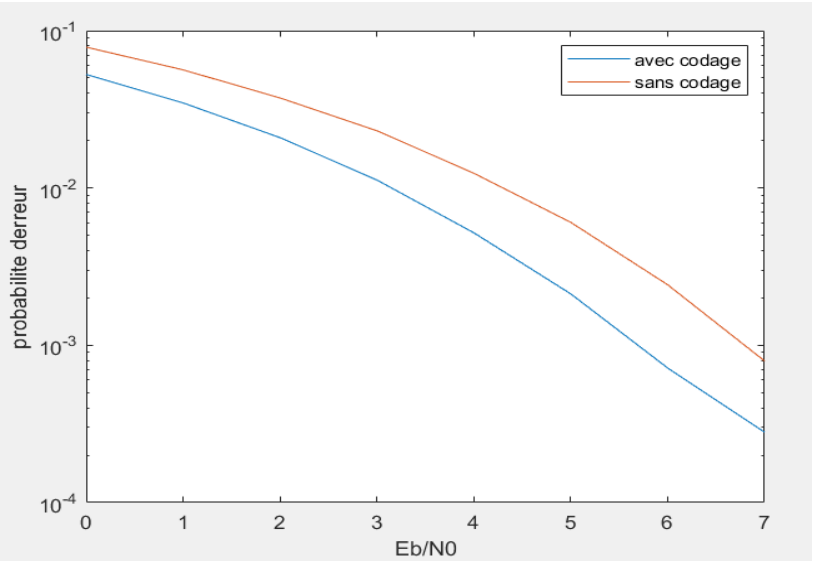
2.1.2. Test avec bruit :

➢ **CODE MATLAB :**

```
%sans codage
Nb=400000;
err=zeros(8);
err1=zeros(8);
P=[0 1 2 3 4 5 6 7];
G=[1 0 0 0 1 1 0; 0 1 0 0 0 1 1; 0 0 1 0 1 1 1; 0 0 0 1 1 0 1];
    for p=0:7
        MSG=randi([0 1], 100000, 4);
        Q=qfunc(2*4/7*sqrt(2*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
        MSG_CODE=encode(MSG_CAN, 7, 4, 'linear/binary', G);
        E=0;
        while E<100
            C=0;
            MSG_RETOUR=decode(MSG_CODE, 7, 4, 'linear/binary', G);
        for i=1:Nb
            if MSG(i) ~= MSG_RETOUR(i)
            C=C+1;
            end
        end
        E=E+C;
        end
    err(p+1)=E/Nb;
    end
err;
%avec codage
    for p=0:7
        MSG=randi([0 1], 100000, 4);
        Q=qfunc(2*4/7*sqrt(2*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
        E=0;
        while E<100
            C=0;
        for i=1:Nb
            if MSG(i) ~= MSG_CAN(i)
            C=C+1;
            end
        end
        E=E+C;
        end
    err1(p+1)=E/Nb;
    end
err1;

semilogy(P, err);
hold on
semilogy(P, err1);
legend('avec codage','sans codage')
```

➢ RESULTATS :



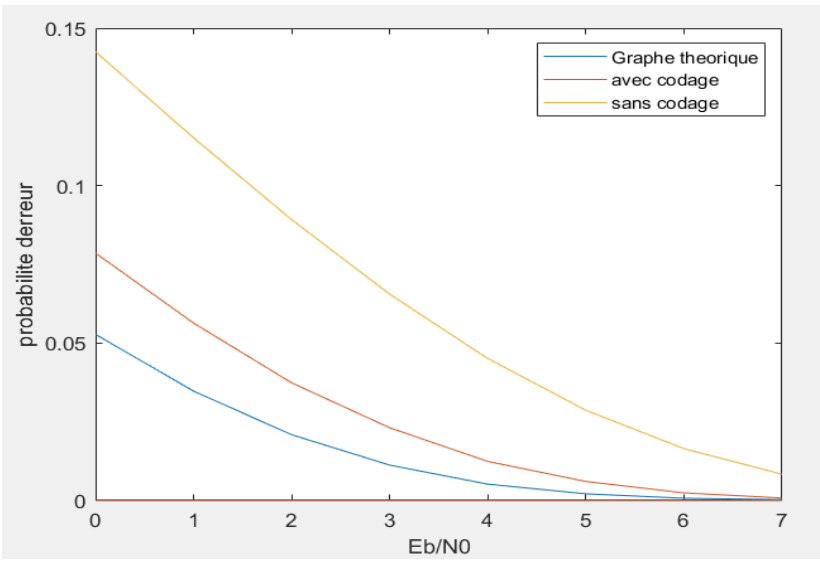


Figure1 **Figure2**

➢ REMARQUES :

- La **figure1 (**tracée dans l'échelle logarithmique avec la fonction **semilogy( ))** représente la probabilité d'erreur en fonction de E_b/N_0 pour les 2 cas, après un codage canal et sans codage canal, on remarque bien que la probabilité d'erreur a diminuée en utilisant un codage correcteur d'erreurs.
- Pour le traçage de la **figure2,** j'ai utilisé la fonction **plot( )** pour pouvoir avoir la courbe théorique, voici le **CODE MATLAB** correspondant :

```
GRAPH_TH=bercoding(P,'block','hard',7,4,1);
plot(P,GRAPH_TH);
hold on
```

2.2.Code cyclique C(7,4)

➢ CODE MATLAB :

```
%sans codage
clc;clear;
Nb=400000;
err=zeros(8);
err1=zeros(8);
P=[0 1 2 3 4 5 6 7];
    for p=0:7
        MSG=randi([0 1], 100000, 4);
        Q=qfunc(sqrt(2*4/7*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
        **MSG_CODE=encode(MSG_CAN, 7, 4, 'cyclic', cyclpoly(7,4));
        E=0;
        while E<100
            C=0;
            **MSG_RETOUR=decode(MSG_CODE, 7, 4, 'cyclic', cyclpoly(7,4),
syndtable(cyclgen(7,cyclpoly(7,4))));
        for i=1:Nb
            if MSG(i) ~= MSG_RETOUR(i)
```

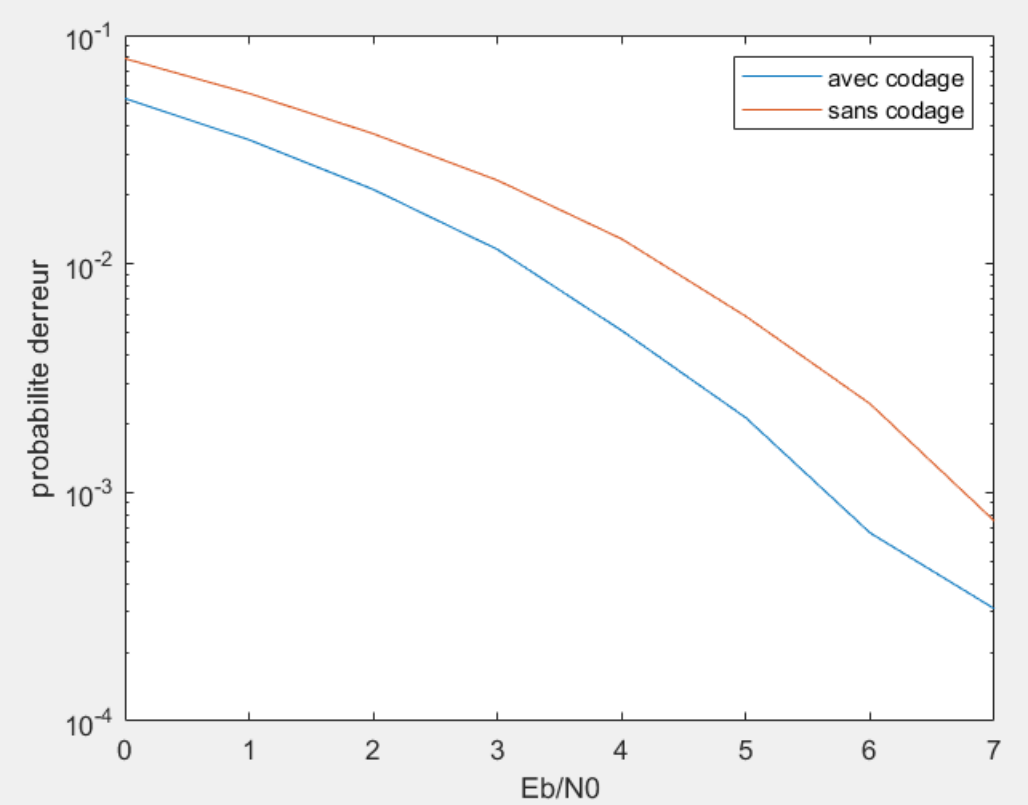
```
            C=C+1;
            end
        end
        E=E+C;
        end
    err(p+1)=E/Nb;
    end
err;
      %avec codage

    for p=0:7
        MSG=randi([0 1], 100000, 4);
        Q=qfunc(sqrt(2*4/7*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
        E=0;
        while E<100
            C=0;
        for i=1:400000
            if MSG(i) ~= MSG_CAN(i)
            C=C+1;
            end
        end
        E=E+C;
        end
    err1(p+1)=E/Nb;
    end
err1;

semilogy(P, err);
hold on
semilogy(P, err1);
legend('avec codage','sans codage');
```

- **RESULTATS :**

- Les lignes à modifier sans précisées par '*'.
- On obtient le graphe :



- Pour le calcul de E_b/N_0, j'ai pensé à utiliser la fonction **biterr( )** qui compare entre les 2 arguments entrées, voici le script correspondant :

```
SANS_CODAGE = biterr(MSG_RETOUR, MSG);
 AVEC_CODAGE = biterr(MSG_CAN, MSG);
if (SANS_CODAGE/400000 > AVEC_CODAGE/400000)
 disp("apres codage est meilleur");
                    end
```

- On aura alors comme résultats :

```
>> AVEC_CODAGE/400000

ans =

    8.2750e-04

>> SANS_CODAGE/400000

ans =

    0.4989
```

2.3.Code convolutif C(3,1/2)

Dans cette partie, j'ai écrit un programme Matlab mais il prend beaucoup de temps pour s'exécuter alors je n'ai pas pu avoir des résultats, voici le **CODE MATLAB :**

```
clc;clear;
trellis = poly2trellis(3, [5 7]);
D= 5;
Nb=100;
err=zeros(8);
err1=zeros(8);
P=[0 1 2 3 4 5 6 7];
    for p=0:7
        MSG=randi([0 1], 1, 100);
        Q=qfunc(sqrt(2*4/7*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
       MSG_CODE = convenc(MSG, trellis);
        E=0;
        while E<5
            C=0;
            MSG_RETOUR=vitdec(MSG_CODE,trellis,D,'trunc','hard');
        for i=1:Nb
            if MSG(i) ~= MSG_RETOUR(i)
            C=C+1;
            end
        end
        E=E+C;
        end
    err(p+1)=E/Nb;
    end
err;
    for p=0:7
        MSG=randi([0 1], 1, 100);
        Q=qfunc(sqrt(2*4/7*10^(p/10)));
        MSG_CAN= bsc(MSG,Q);
        E=0;
        while E<5
            C=0;
        for i=1:100
```

```
            if MSG(i) ~= MSG_CAN(i)
            C=C+1;
            end
        end
        E=E+C;
        end
    err1(p+1)=E/Nb;
    end
err1;

plot(P, err);
hold on
plot(P, err1);
legend('avec codage','sans codage');
```