Ecole Supérieure des Communications de Tunis (SUP'COM)

TP 1

Outils pour la simulation de chaînes de communications numériques

Comptes-rendus effectués par :

Chouchen Rana

Koubaa Adem

Gormazi Ahmed

Laamouri Karim

INDP2E

2024/2025

I- Objectif :

Le but de ce TP est de se familiariser avec quelques notions essentielles de communications numeriques par le biais de la simulation sous Matlab.

II- Introduction :

Dans ce TP, l'objectif est de se familiariser avec les notions fondamentales des communications numériques en utilisant une simulation sous Matlab. Plus précisément, nous étudions la transmission d'une suite de symboles d'information à travers un canal de type BBAG (Bruit Blanc Additif Gaussien). La chaîne de transmission est composée de plusieurs blocs, incluant une modulation MDA et un filtrage analogique linéaire. Le signal émis, après modulation et mise en forme, est soumis à un bruit additif sur le canal, et nous analysons son impact sur la réception du signal. Ce TP permet de comprendre et de simuler les processus clés de la transmission numérique en présence de bruit.

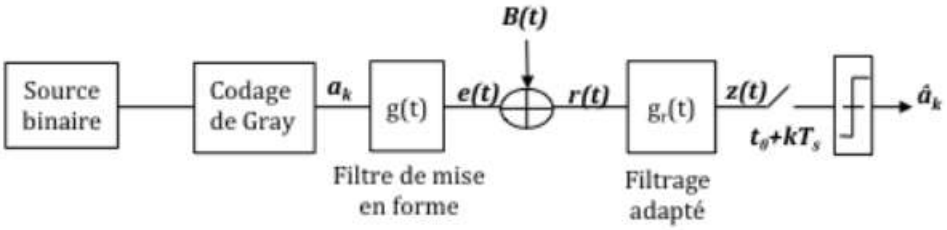


FIGURE 1 – Chaîne de transmission numérique utilisant une modulation MDA

III- Travail demandé :

Partie I :

Question 1 :

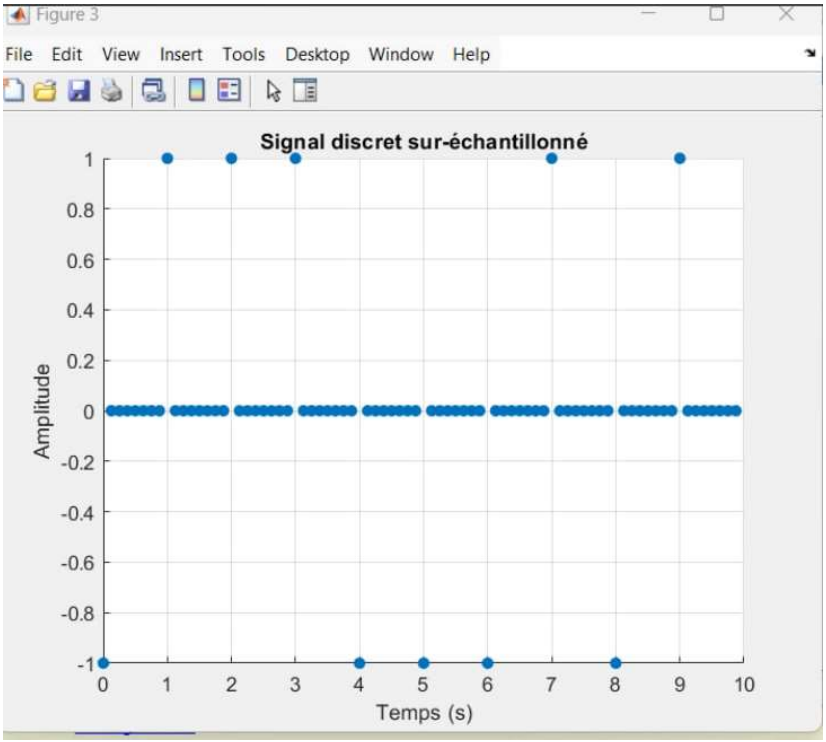
```
% Partie I
% qeustion1
N = 10;
source_binaire = randi([0, 1], 1, N);
```

Question 2 :

```
% qeustion2
A = 1;
ak = A*(2 * source_binaire - 1);
F = 8;
Tb = 1;
signal_discret = zeros(1, F * N);
signal_discret(1:F:end) = ak;

t = (0:length(signal_discret)-1) * (Tb / F);
```

Question 3 :

```
% qeustion3
figure;
scatter(t, signal_discret, 'filled');
xlabel('Temps (s)');
ylabel('Amplitude');
title('Signal discret sur-échantillonné');
grid on;
```



Partie II :

Question 1 :

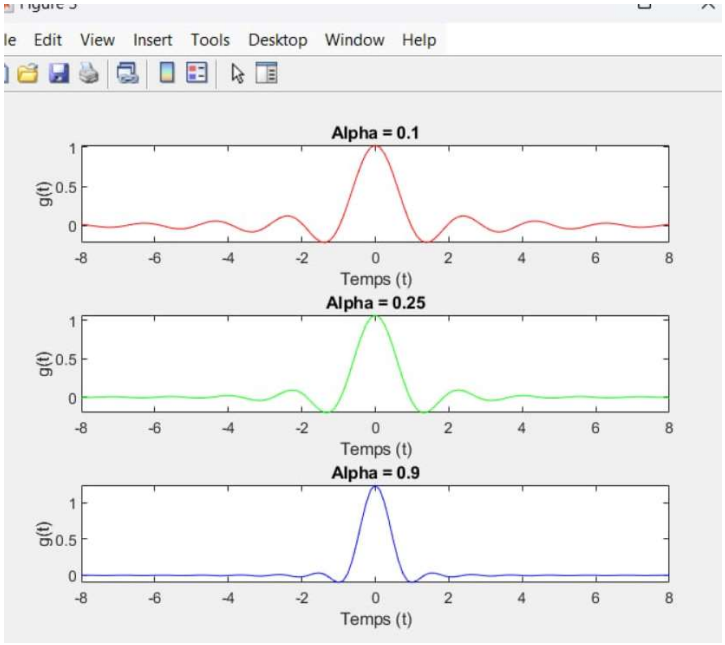
```
% Partie 2
%question1:
Tb=1;
syms t;
alpha1=0.1;
alpha2=0.25;
alpha3=0.9;

f1=g(t,Tb,alpha1) ; %calculer g(t,Tb,0.1
f2=g(t,Tb,alpha2) ;
f3=g(t,Tb,alpha3) ;
%cas alpha=0.1
Lim11=limit (f1,t,0);
Lim12=limit(f1,t,(Tb/(4*alpha1)));
Lim13=limit(f1,t,(-Tb/(4*alpha1)));
%cas alpha=0.25
Lim21=limit (f2,t,0);
Lim22=limit(f2,t,(Tb/(4*alpha2)));
Lim23=limit(f2,t,(-Tb/(4*alpha2)));
%cas alpha=0.9
Lim31=limit (f3,t,0);
Lim32=limit(f3,t,(Tb/(4*alpha3)));
Lim33=limit(f3,t,(-Tb/(4*alpha3)));
```

Question 2 :

```
%question2:
K=8;
Tb=1;
axe_x=[-K*Tb:1/8:K*Tb];

A1=[];
A2=[];
A3=[];

for i=axe_x;
 A1=[A1 eval(limit(g(t,Tb,alpha1),t,i))];
 A2=[A2 eval(limit(g(t,Tb,alpha2),t,i))];
 A3=[A3 eval(limit(g(t,Tb,alpha3),t,i))];
```

Question 3 :

```
%question3:

figure;
% Sous-graphe pour alpha = 0.1
subplot(3, 1, 1);
plot(axe_x, A1, 'r');
title('Alpha = 0.1');
xlabel('Temps (t)');
ylabel('g(t)');

% Sous-graphe pour alpha = 0.25
subplot(3, 1, 2);
plot(axe_x, A2, 'g');
title('Alpha = 0.25');
xlabel('Temps (t)');
ylabel('g(t)');

% Sous-graphe pour alpha = 0.9
subplot(3, 1, 3);
plot(axe_x, A3, 'b');
title('Alpha = 0.9');
xlabel('Temps (t)');
ylabel('g(t)');
```



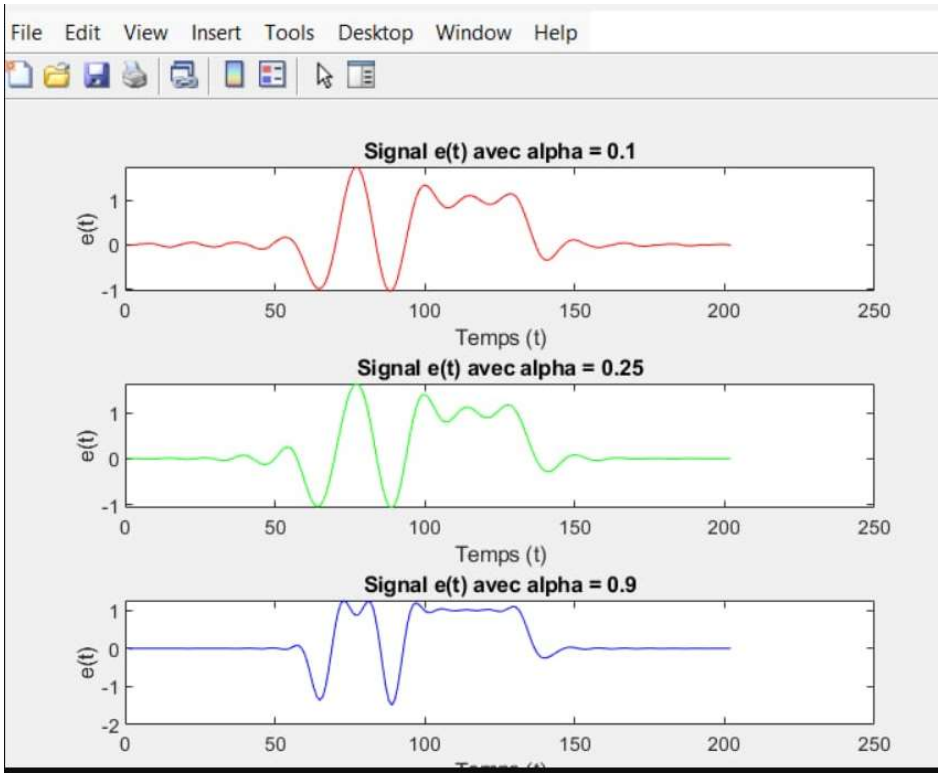
Question 4:

```
%question4:
% Calcul de l'énergie pour chaque cas
disp('énergie dans le cas où alpha=0.1');
Energie1 = sum(A1.^2);  % Calcul de l'énergie pour alpha = 0.1
disp(Energie1);

disp('énergie dans le cas où alpha=0.25');
Energie2 = sum(A2.^2);  % Calcul de l'énergie pour alpha = 0.25
disp(Energie2);

disp('énergie dans le cas où alpha=0.9');
Energie3 = sum(A3.^2);  % Calcul de l'énergie pour alpha = 0.9
disp(Energie3);
```

```
énergie dans le cas où alpha=0.1
    7.9955

énergie dans le cas où alpha=0.25
    7.9995

énergie dans le cas où alpha=0.9
    8.0000
```

Question 5:

```
%question5:
at=[];
for i=1:(length(ak)-1)
    at=[at ak(i) zeros(1,F-1)];
end
at=[Se ak(length(ak))];
e1=conv(at,A1);
e2=conv(at,A2);
e3=conv(at,A3);
% Affichage des résultats avec des sous-graphes
figure;

% Sous-graphe pour e1 (alpha = 0.1)
subplot(3, 1, 1);
plot(e1, 'r');
title('Signal e(t) avec alpha = 0.1');
xlabel('Temps (t)');
ylabel('e(t)');

% Sous-graphe pour e2 (alpha = 0.25)
subplot(3, 1, 2);
plot(e2, 'g');
title('Signal e(t) avec alpha = 0.25');
xlabel('Temps (t)');
ylabel('e(t)');

% Sous-graphe pour e3 (alpha = 0.9)
subplot(3, 1, 3);
plot(e3, 'b');
title('Signal e(t) avec alpha = 0.9');
xlabel('Temps (t)');
ylabel('e(t)');
```



Partie III :

Question 1 : Etudier la fonction “randn” de Matlab. Deduire comment on peut générer un bruit Gaussien de moyenne nulle et de variance σ 2 quelconque ?

La fonction randn de Matlab génère des nombres aléatoires suivant une distribution normale standard, c'est-à-dire une distribution gaussienne de **moyenne nulle** et de **variance égale à 1**.

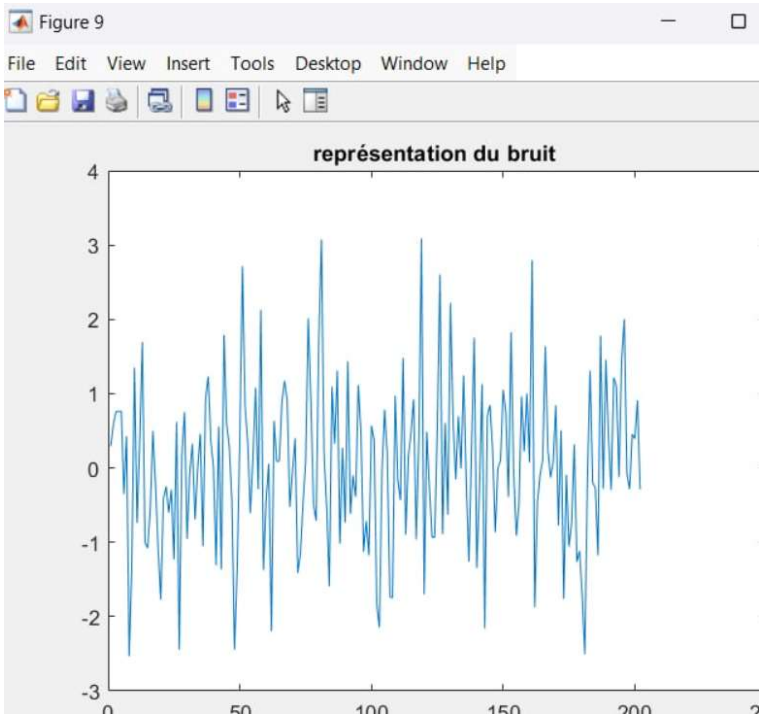
Pour générer un bruit gaussien de moyenne nulle et de variance **σ^2** (différente de 1), il suffit de **scaler** le bruit généré par randn. En effet, si on multiplie le bruit standard randn par un facteur de σ, le résultat sera un bruit gaussien de variance σ^2.

Plus précisément :

Si **x∼N(0,1)** (c'est-à-dire que x suit une distribution normale de moyenne nulle et de variance 1), alors **σx** suit une distribution **N(0,σ2),** c'est-à-dire une normale de moyenne nulle et de variance σ2.

Question 2 :

```
%Partie3 question2
%cas ou RSB en db =5
RSB=5; % en db
Eb=F;
N0=Eb*10^(-RSB/10);
sigma=sqrt(N0/2);
B=sigma*randn(1,length(e1)); %  générer un Bruit blanc Gaussien
figure;
plot(B);
title('représentation du bruit ');
Pe=1/2*erfc(sqrt(Eb/N0));
disp(Pe)
```

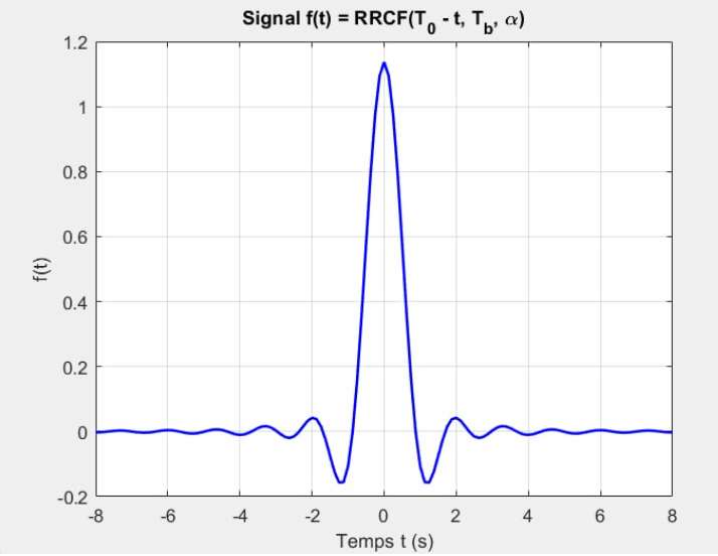


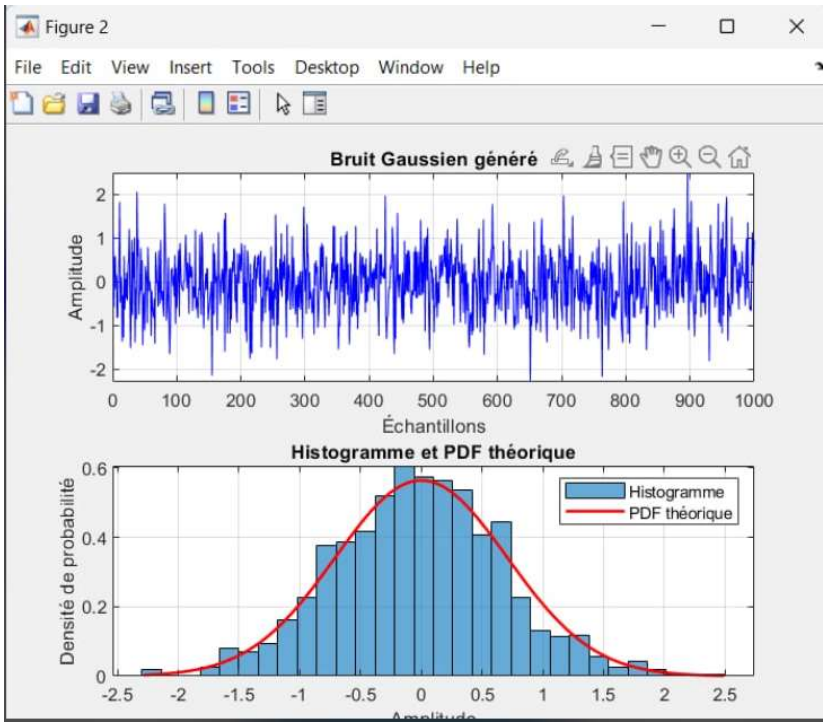

```
énergie dans le cas où a
    8.0000

probabilite erreur:
    0.0060
```

Partie IV :

Question 1 : **Déterminer le filtre adapté**

```
% Paramètres
T0 = 5;               % Temps décalé
Tb = 1;               % Temps symbole
k = 8;                % Paramètre pour l'intervalle de temps
ts = 1;               % Pas de temps
alpha = 0.5;          % Paramètre de modulation
t = -k*ts : ts/8 : k*ts;  % Vecteur de temps

% Initialisation du vecteur f
f = zeros(1, length(t));

% Boucle pour calculer f(t) = g(T0 - t, Tb, alpha)
for i = 1:length(t)
    % Appel de g(t) pour chaque valeur scalaire de t
    f(i) = RRCF(T0 - t(i), Tb, alpha);  % Calcul de f(t) pour t(i)
end

% Affichage des résultats
figure;
plot(t, f, 'b', 'LineWidth', 1.5);
xlabel('Temps t (s)');
ylabel('f(t)');
title('Signal f(t) = g(T_0 - t, T_b, \alpha)');
grid on;

% Définition de la fonction g(t, Tb, alpha)
function y = g(t, Tb, alpha)
    % Exemple de fonction g(t) (vous pouvez la remplacer par votre propre fonction)
    y = sinc(t / Tb) * cos(2 * pi * alpha * t);
end
```





Question 2 :

```
%question2
% Définir les paramètres pour chaque cas
alphas = [0.25, 0.5, 0.9]; % Valeurs de alpha
e_values = {e1, e2, e3};   % Cellule contenant les vecteurs e
A_values = {A1, A2, A3};   % Cellule contenant les vecteurs A
for k = 1:length(alphas)
    alpha = alphas(k);
    e = e_values{k};
    A = A_values{k};
    % Convolution
    y_t = conv(e, A, 'same');
    % Affichage du diagramme de l'œil
    figure;
    hold on;
    for n = 1:N-1
        plot((-F:F-1)/F, y_t((n-1)*F+1:(n+1)*F), 'b'); % Superposer les traces
    end
    title(['Diagramme de l'œil pour \alpha = ', num2str(alpha)]);
    xlabel('Temps (échantillons normalisés)');
    ylabel('Amplitude');
    grid on;
    hold off;
end
```

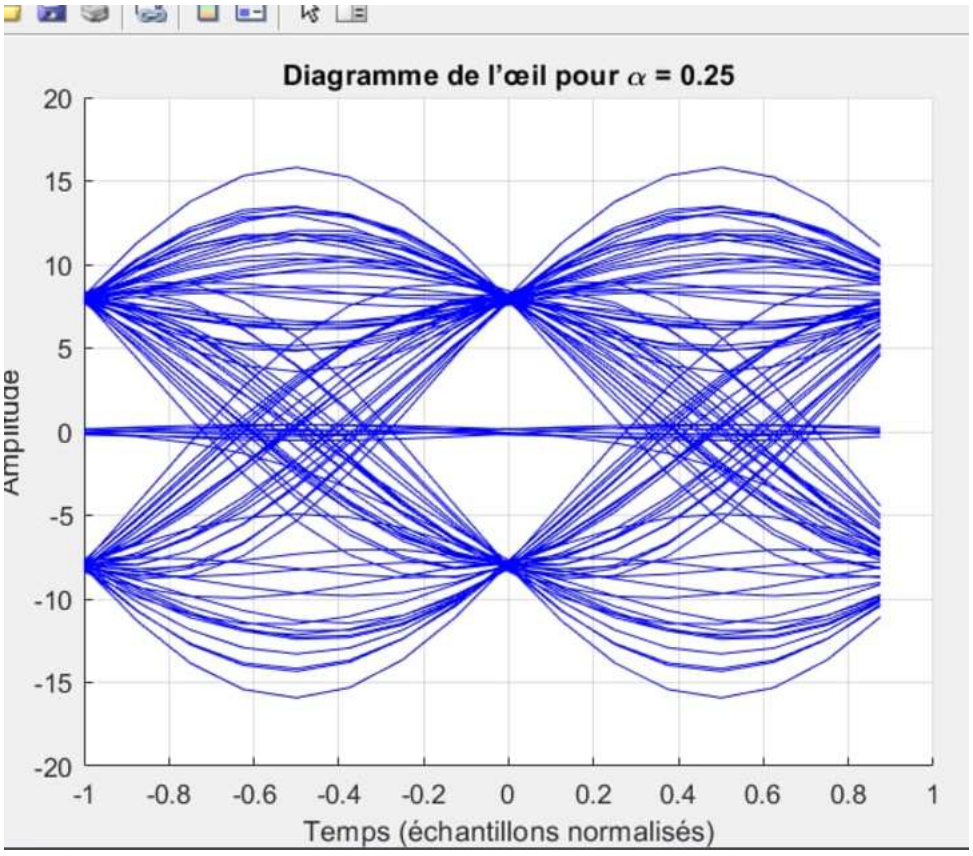


Diagramme de l'œil pour α = 0.25
Amplitude
20
15
10
5
0
-5
-10
-15
-20
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Temps (échantillons normalisés)

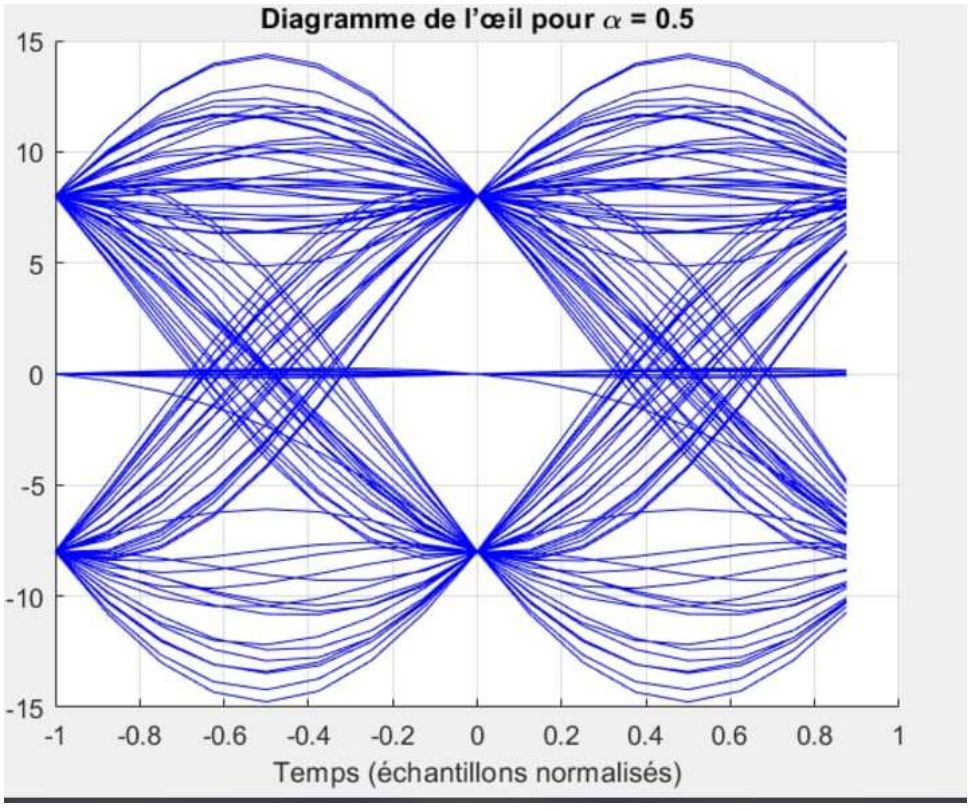
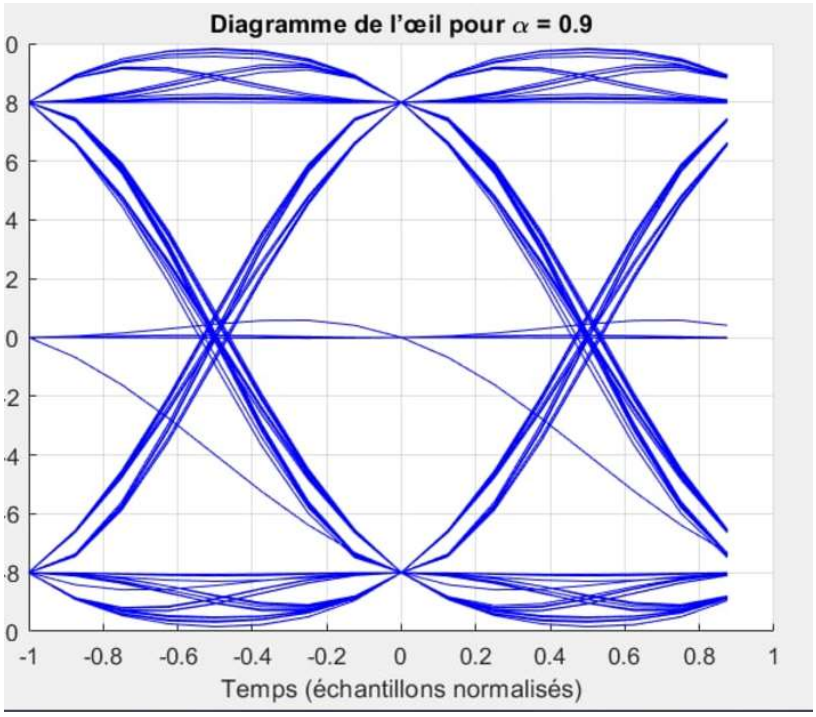


Diagramme de l'œil pour α = 0.5
15
10
5
0
-5
-10
-15
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Temps (échantillons normalisés)



Question 3:

```
%question 3
K_values = [2, 4];
for K = K_values
    axe_x = -K*Tb : 1/8 : K*Tb;
    at = [];
    for i = 1:(length(ak) - 1)
        at = [at ak(i) zeros(1, F - 1)];
    end
    at = [at ak(end)];

    A1 = arrayfun(@(t) g(t, Tb, alpha1), axe_x);

    e1 = conv(at, A1);
    y_t = conv(e1, A1, 'same');

    figure;
    hold on;
    for n = 1:N-1
        plot((-F:F-1)/F, y_t((n-1)*F+1:(n+1)*F), 'b'); % Superposer les traces
    end
    title(['Diagramme de l'œil pour K = ', num2str(K)]);
    xlabel('Temps (échantillons normalisés)');
    ylabel('Amplitude');
    grid on;
    hold off;
end
```

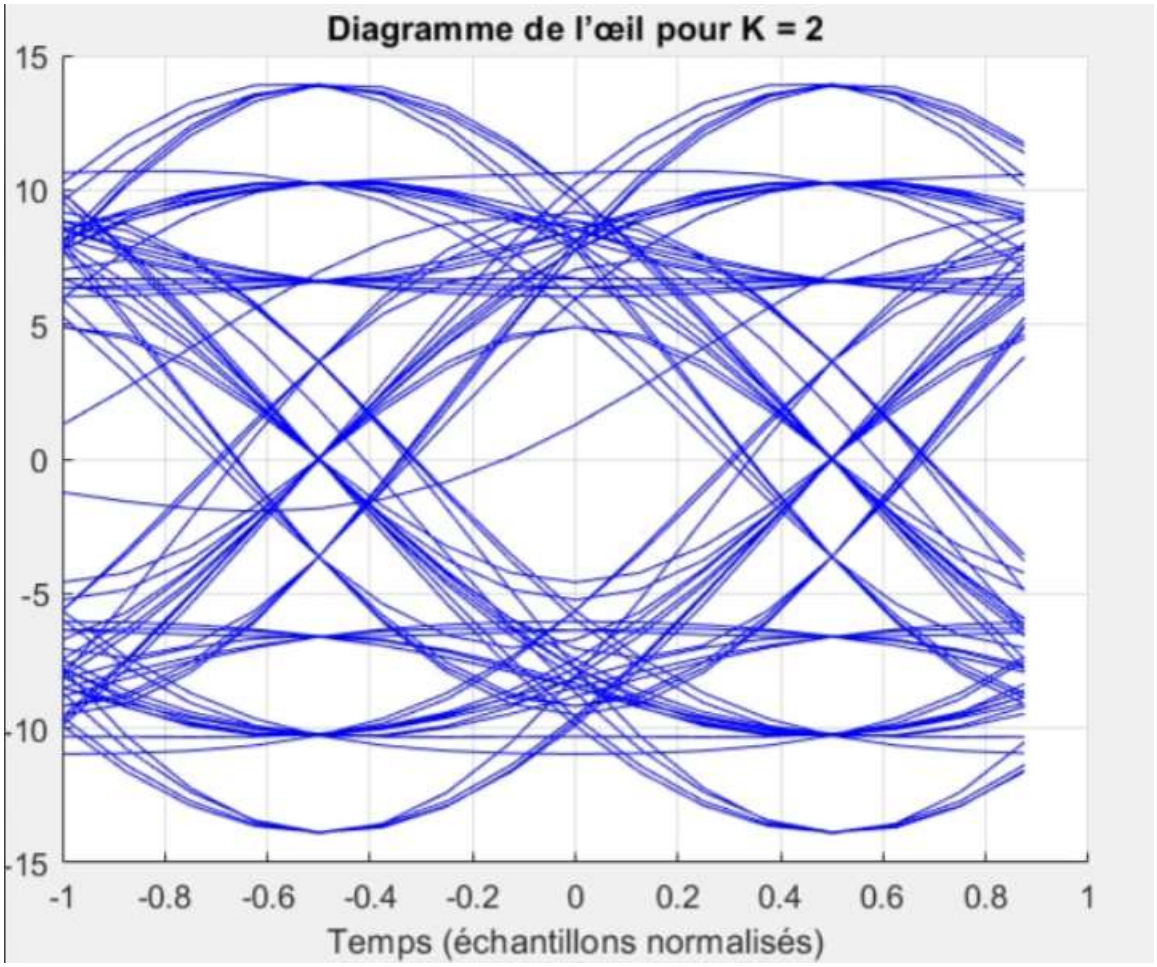


Diagramme de l'œil pour K = 2
15
10
5
0
-5
-10
-15
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Temps (échantillons normalisés)

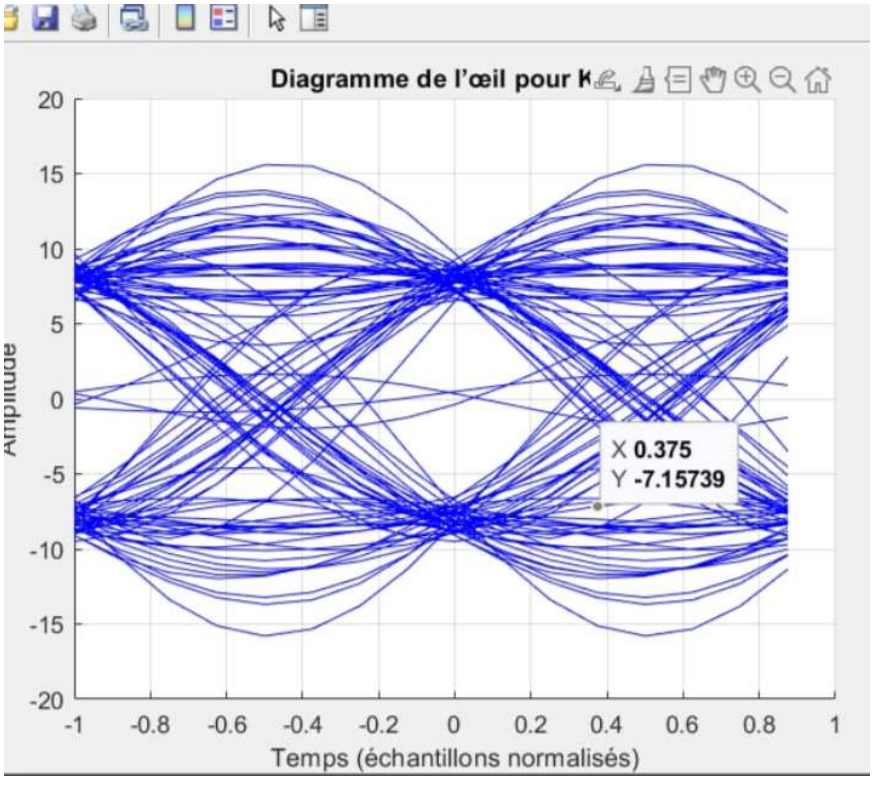
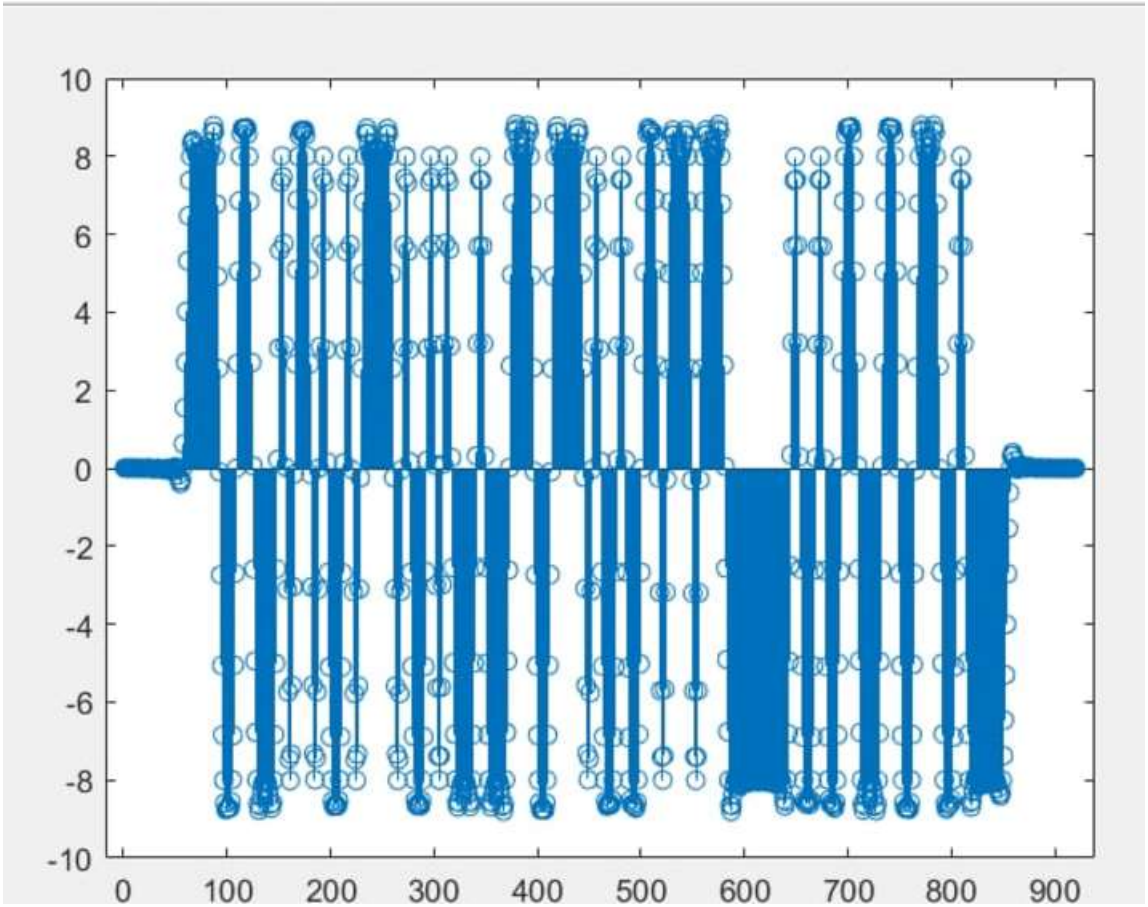


Diagramme de l'œil pour K
20
15
10
5
0
-5
-10
-15
-20
X 0.375
Y -7.15739
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Temps (échantillons normalisés)

Question 4

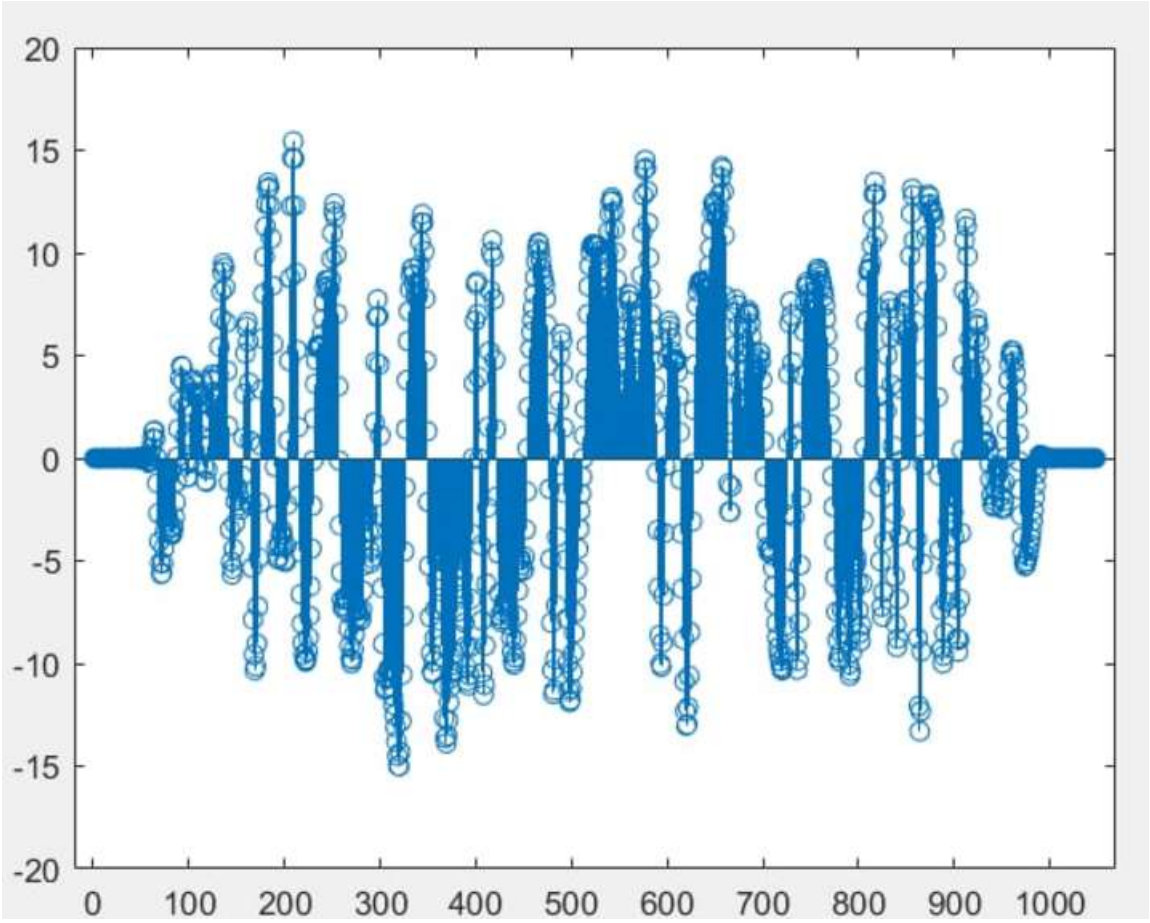
```
%question4

y_t = conv(e3, A3, 'same');
% Affichage du diagramme de l'œil
figure;
hold on;
for n = 1:N-1
plot((-F:F-1)/F, y_t((n-1)*F+1:(n+1)*F), 'b'); % Superposer les traces
end
title(['Diagramme de l'œil pour \alpha = ', num2str(alpha)]);
xlabel('Temps (échantillons normalisés)');
ylabel('Amplitude');
grid on;
hold off;
stem(y_t)
```



En peut voir d'Apres la courbe que z(t0+nT)=ak*F=8 si ak=1 ou -8 si ak=-1 avec F =8

Question 5

```
%question5
B = randn (1, length(e3)) % Bruit blanc Gaussien
RSB = 5; %en dB
N0 = Eb * 10.^(-RSB / 10);
B = sqrt (N0/2)*B
y_t=conv(e3+B,A3)
figure;
hold on;
for n = 1:N-1
    plot((-F:F-1)/F, y_t((n-1)*F+1:(n+1)*F), 'b'); % Superposer les traces
end
title('Diagramme de l'œil');
xlabel('Temps (échantillons normalisés)');
ylabel('Amplitude');
grid on;
hold off;
stem(y_t)
```



Dans cette simulation, l'ajout de bruit blanc gaussien au signal engendre une dégradation de la qualité du signal observé. Cela peut poser des problèmes au moment de la prise de décision, car le bruit peut masquer les informations utiles, rendant plus difficile l'identification correcte des symboles ou des états

du signal. En conséquence, le système peut commettre davantage d'erreurs lors de la démodulation ou du décodage.

IV- Conclusion

Ce TP a permis de comprendre et de simuler les principaux aspects d'une chaîne de transmission numérique en présence de bruit. Nous avons généré une source binaire, appliqué un codage de Gray, puis simulé l'émission et la réception de symboles à travers un canal soumis à un bruit blanc additif gaussien (BBAG). L'utilisation de la fonction randn pour générer ce bruit, et la simulation du filtre de mise en forme en racine de cosinus surélevé, ont été des étapes clés dans l'étude de l'impact du bruit sur la transmission des symboles. L'échantillonnage du signal à une fréquence plus élevée (suréchantillonnage), la convolution pour obtenir le signal émis et la vérification de l'énergie transmise ont permis de simuler le comportement réel d'une transmission numérique.

En analysant les diagrammes de l'œil, nous avons observé l'effet du coefficient de retombée sur l'ouverture horizontale, ce qui a permis de mieux comprendre l'importance de la mise en forme des signaux dans les systèmes de communication. La simulation des filtres et l'étude de l'interférence entre symboles ont montré les compromis entre performance et complexité dans la conception des systèmes de communication. Enfin, les calculs de la variance du bruit et les simulations du filtre adapté ont permis de valider la chaîne de transmission en l'absence de bruit et de vérifier la précision des décisions prises sur les symboles reçus.

Les résultats expérimentaux ont montré que l'efficacité du système dépend fortement des paramètres choisis, comme le coefficient de retombée du filtre, et ont permis d'illustrer la manière dont les techniques de mise en forme et de filtrage contribuent à optimiser les performances de transmission dans un environnement bruyant.