Table of Contents

```
clear all;
close all;
clc;
```

Devoir 4

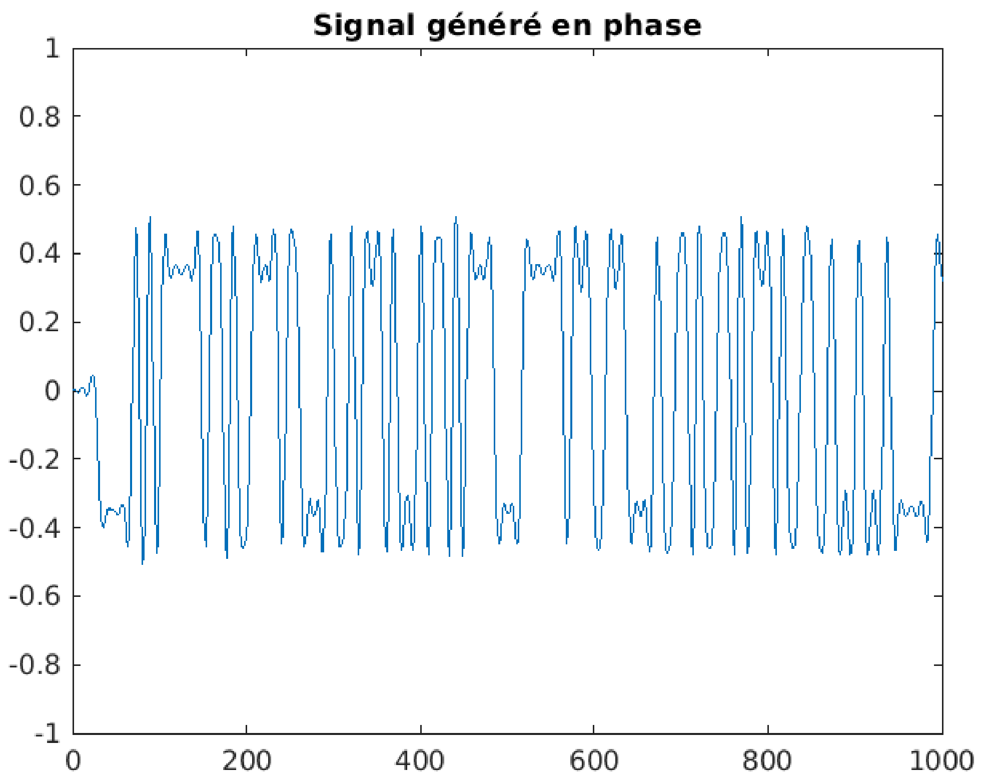
Mahmoud LAANAIYA

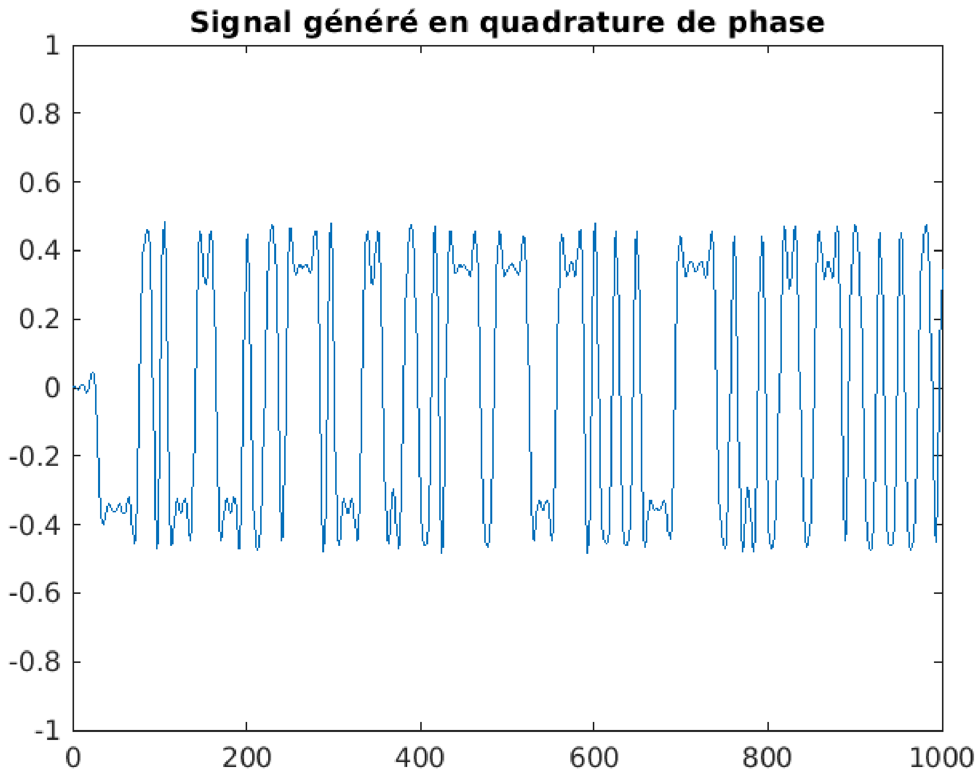
Première Partie

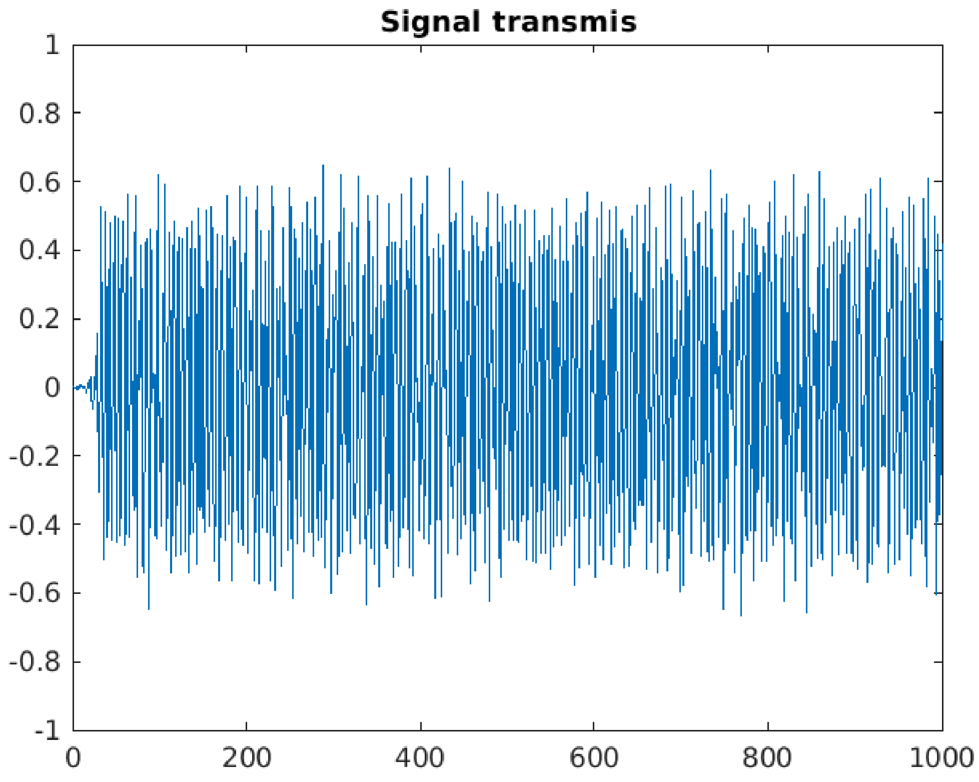
```
Donnees = randi([0 1], 1, 10000);
% Implantation de la chaine sur fréquence porteuse
Rb = 2000;
Fe = 10000;
Te = 1/Fe;
alpha = 0.8;
fp = 2000;
Ns = 8;
M = 4;
eb_N0_db = [0 : 8];
ak = 2*Donnees(1:2:end) - 1;
bk = 2*Donnees(2:2:end) - 1;
mapping = ak + j*bk;
h = rcosdesign(0.5,8,Ns); % Filtre en cosinus
Kronr = kron(mapping, [1 zeros(1, Ns-1)]); % Positionnement des zéros
 entre les ak
xe = filter(h, 1, [Kronr zeros(1,Ns*8)]); % Filtrage avec filtre en
 cosinus
xe_real = real(xe);
figure, plot(xe_real);
axis([0 1000 -1 1]);
title('Signal généré en phase');
xe_im = imag(xe);
figure, plot(xe_im);
axis([0 1000 -1 1]);
title('Signal généré en quadrature de phase');
Nombre_Tot = Ns * (length(Donnees)+16)/2;
t = 0 : Te : (Nombre_Tot-1)*Te;
```

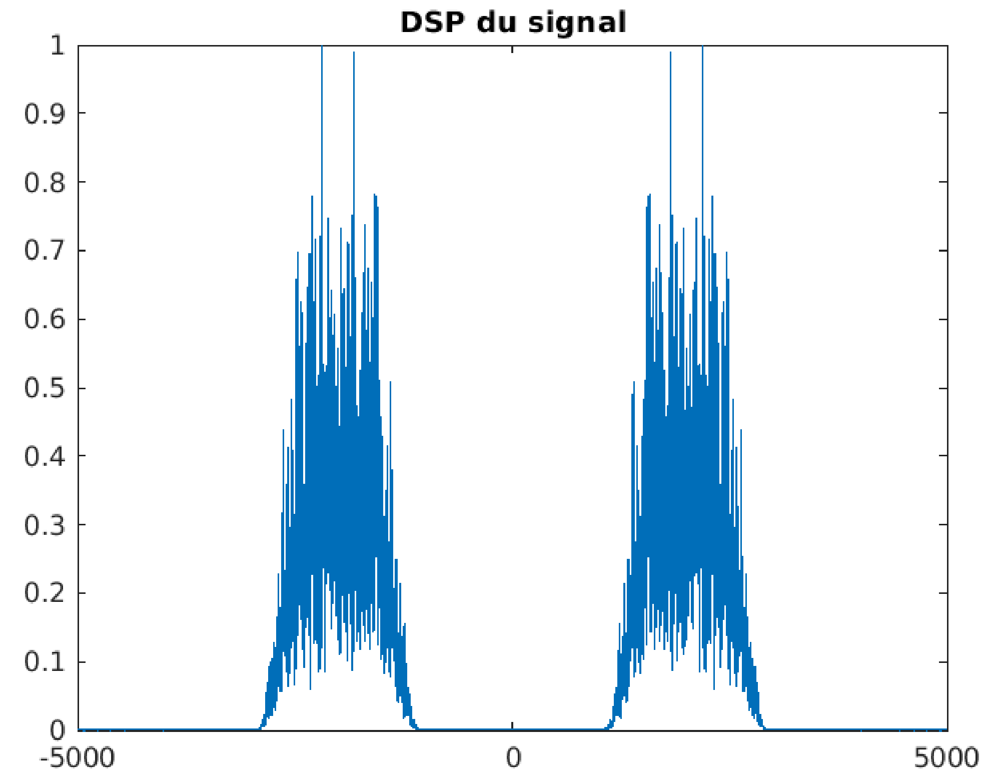
```
expo = exp(j*2*pi*fp*t);
x = real(xe.*expo);
figure, plot(x);
axis([0 1000 -1 1]);
title('Signal transmis');
DSP_x = pwelch(x, [], [], Fe, "centered"); % Densité spectrale de
 puissance
DSP_x = DSP_x./max(DSP_x); % Normaliser la DSP
frequence = -Fe/2:Fe/(length(DSP_x)-1):Fe/2; % Interval de fréquence
 discret
figure, plot(frequence, DSP_x);
title("DSP du signal");
xcos = x.*cos(2*pi*fp*t);
xsin = -j*x.*sin(2*pi*fp*t);
hr = fliplr(h);
X = filter(hr, 1, xsin+xcos);
X_ech = X(8*Ns+1:Ns:end);
dec_reel = sign(real(X_ech));
dec_imaginaire = sign(imag(X_ech));
demapping = (dec_reel+1 + j*(dec_imaginaire+1))/2;
dem = zeros(1, length(Donnees));
dem(1:2:end) = real(demapping);
dem(2:2:end) = imag(demapping);
test = Donnees - dem ;
TEB = length(find(test~=0))/length(Donnees);

% Implantation de la chaine avec bruit
tmp = zeros(1,length(eb_N0_db));
precision = 100;
for i = 1 : precision
    Donnees = randi([0 1], 1, 10000);
    for k = 1 : length(eb_N0_db)
        ak = 2*Donnees(1:2:end) - 1;
        bk = 2*Donnees(2:2:end) - 1;
        mapping = ak + j*bk;
        Kronr = kron(mapping, [1 zeros(1, Ns-1)]); % Positionnement
 des zéros entre les ak
        xe = filter(h, 1, [Kronr zeros(1,Ns*8)]); % Filtrage avec
 filtre en cosinus
        t = (0:length(xe)-1)*Te;
        expo = exp(j*2*pi*fp*t);
        x = real(xe.*expo);
        Px = mean(abs(x).^2);
        sigma2 = Px*Ns/(2*log2(M)*10^(eb_N0_db(k)/10));
        bruit = sqrt(sigma2)*randn(1, length(x));
        Z = x + bruit;
        Zcos = Z.*cos(2*pi*fp*t);
        Zsin = -j*Z.*sin(2*pi*fp*t);
        z = filter(h, 1, Zcos+Zsin); % Signal filtré
        z_ech = z(8*Ns+1:Ns:end); % Échantillonage
        % Decision
        dec_reel = sign(real(z_ech));
        dec_imaginaire = sign(imag(z_ech));
        demapping = (dec_reel+1 + j*(dec_imaginaire+1))/2;
```

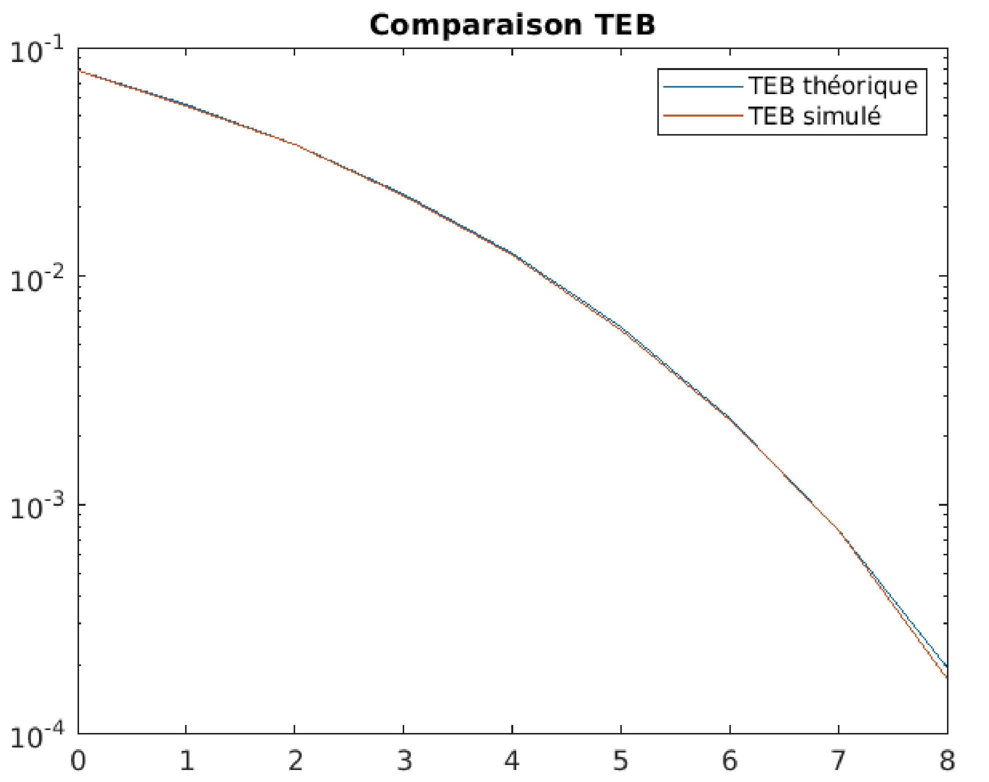
```
            bits = zeros(1, length(Donnees));
            bits(1:2:end) = real(demapping);
            bits(2:2:end) = imag(demapping);
            test = bits - Donnees;
            TEB_bruit(k) = length(find(test~=0))/length(Donnees);
        end
        tmp = tmp + TEB_bruit;
    end
    TEB_bruit = tmp/precision;
    TEB_th = qfunc(sqrt(2*10.^(eb_N0_db/10)));
    figure();
    semilogy(eb_N0_db, TEB_th, 'DisplayName', 'TEB théorique');hold on
    semilogy(eb_N0_db, TEB_bruit, 'DisplayName', 'TEB simulé');
    legend;
    title('Comparaison TEB');
    hold off
    figure();
    scatterplot(mapping);
    title("Les constellations en sortie du mapping");
    scatterplot(z_ech);
    title("Les constellations en sortie de l'échantillonneur");
```

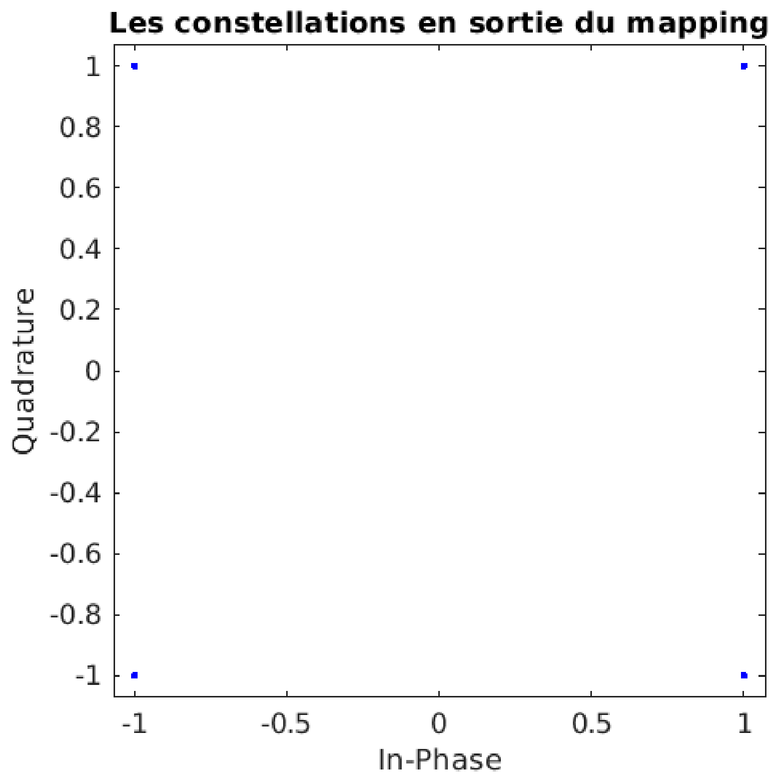


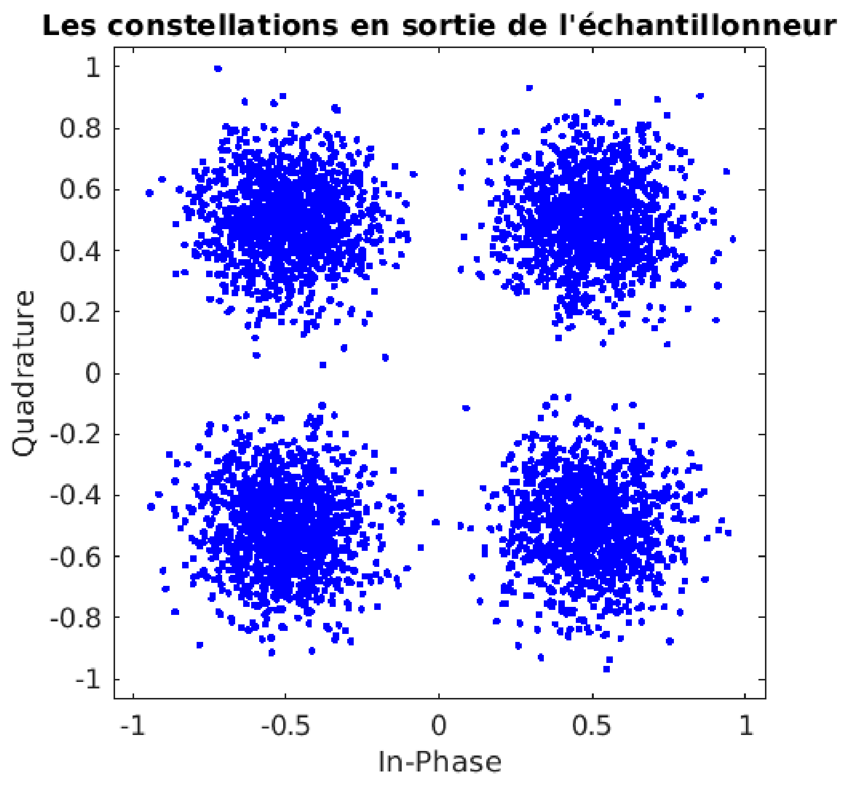

Signal généré en quadrature de phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000


Signal transmis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000


DSP du signal
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5000
0
5000


Comparaison TEB
TEB théorique
TEB simulé
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
7
8

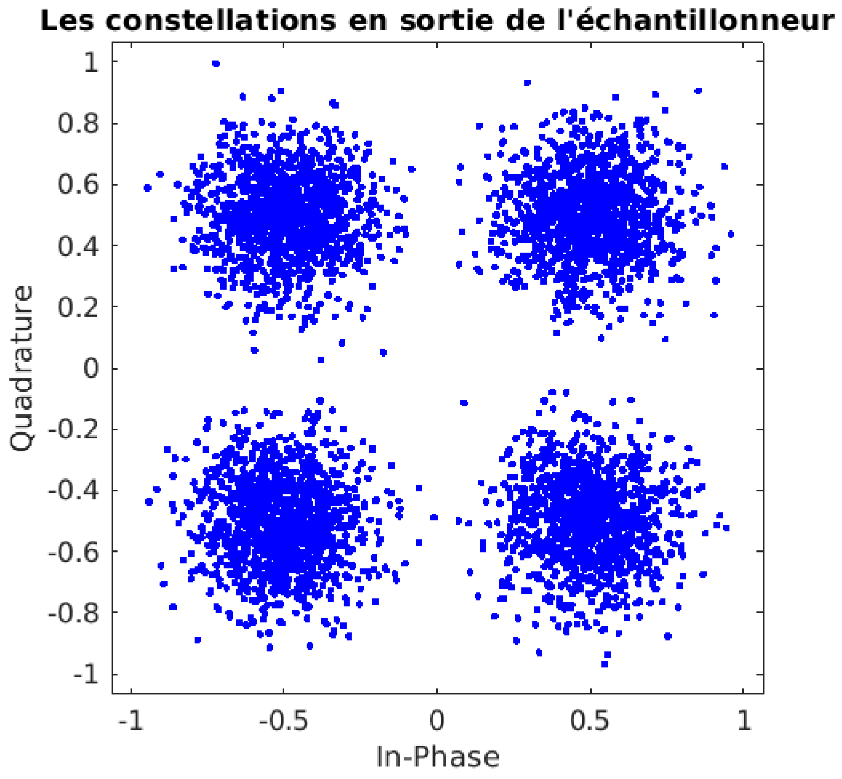

Les constellations en sortie du mapping
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

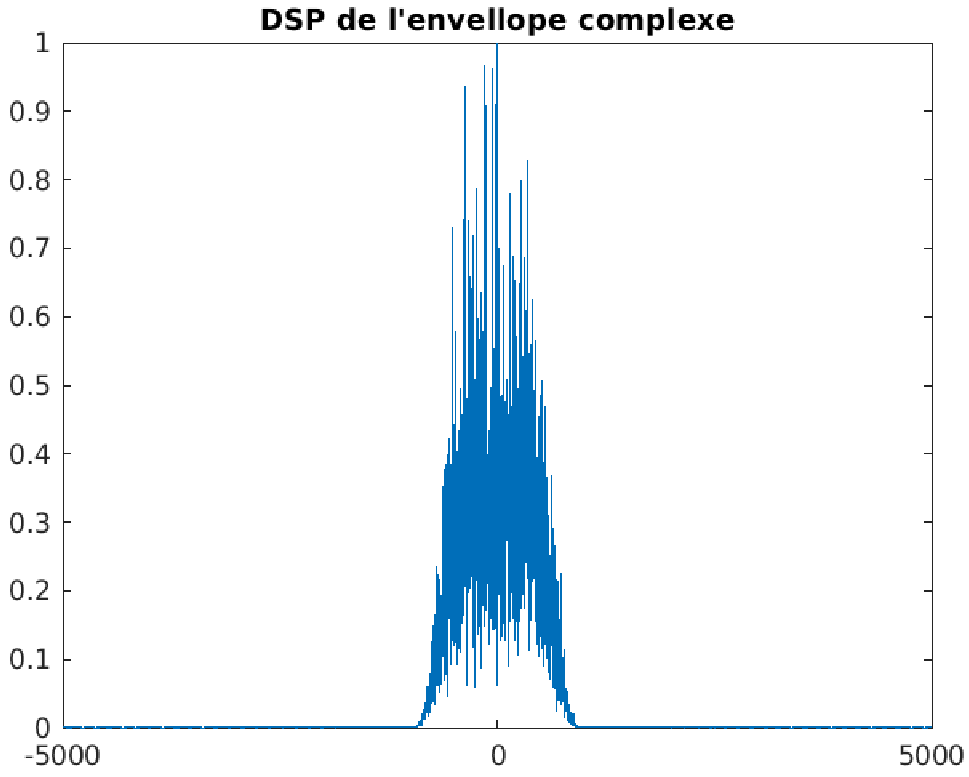


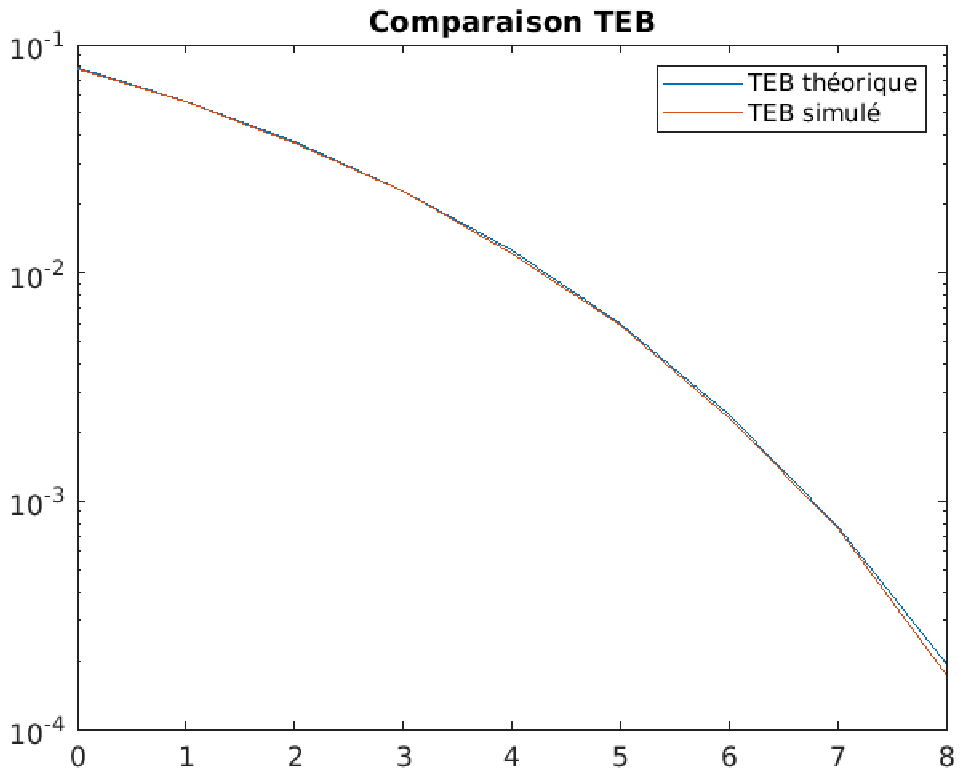
Deuxième Partie

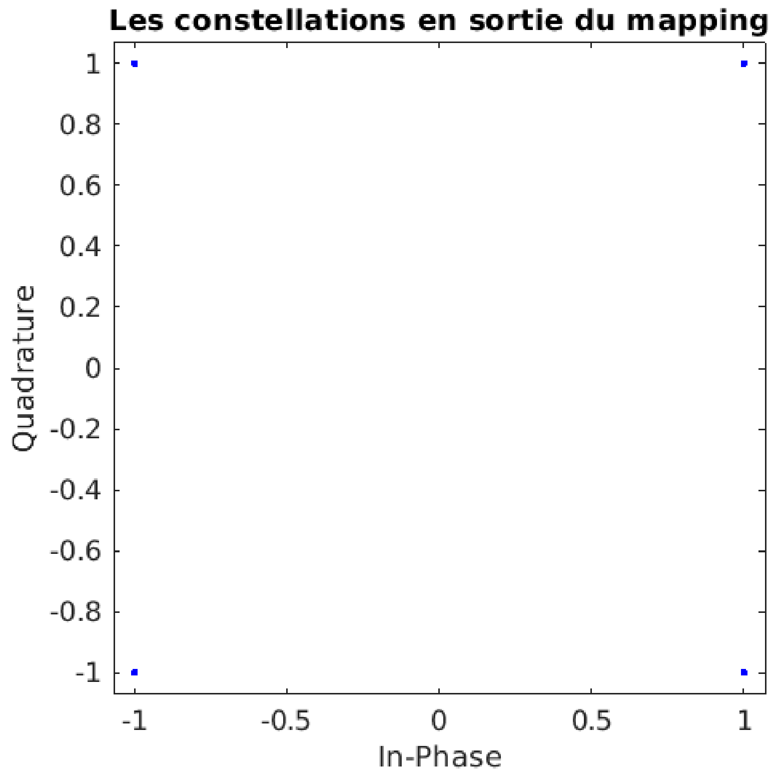
```
mapping = ak + j*bk;
h = rcosdesign(0.5,8,Ns); % Filtre en cosinus
Kronr = kron(mapping, [1 zeros(1, Ns-1)]); % Positionnement des zéros
 entre les ak
xe = filter(h, 1, [Kronr zeros(1,Ns*8)]); % Filtrage avec filtre en
 cosinus
DSP_xe = pwelch(xe, [], [], Fe, "centered"); % Densité spectrale de
 puissance
DSP_xe = DSP_xe./max(DSP_xe); % Normaliser la DSP
figure, plot(frequence, DSP_xe);
title("DSP de l'envellope complexe");
Nombre_Tot = Ns * (length(bits)+16)/2;
t = 0 : Te : (Nombre_Tot-1)*Te;
hr = fliplr(h);
Xe = filter(hr, 1, xe);
X_eche = Xe(8*Ns+1:Ns:end);
dec_reele = sign(real(X_eche));
dec_imaginairee = sign(imag(X_eche));
demappinge = (dec_reele+1 + j*(dec_imaginairee+1))/2;
deme = zeros(1, length(bits));
deme(1:2:end) = real(demappinge);
deme(2:2:end) = imag(demappinge);
teste = bits - deme;
```

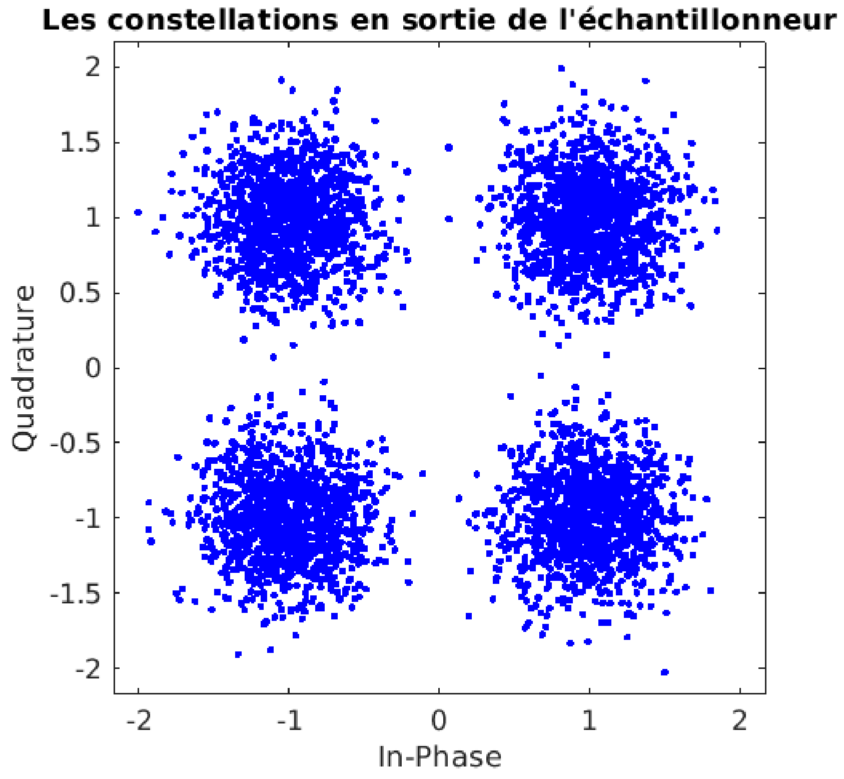
```
TEBe = length(find(teste~=0))/length(bits);
Pxe = mean(abs(xe).^2);
tmpe = zeros(1,length(eb_N0_db));
for k = 1:precision
    Donneese = randi([0 1], 1, 10000);
    for i = 1 : length(eb_N0_db)
        sigma2 = Pxe*Ns/(2*log2(M)*10^(eb_N0_db(i)/10));
        bruitr = sqrt(sigma2)*randn(1, length(real(xe)));
        bruiti = sqrt(sigma2)*randn(1, length(imag(xe)));
        bruite = bruitr + j*bruiti;
        Ze = xe + bruite;
        ze = filter(hr, 1, Ze); % Signal filtré
        z_eche = ze(8*Ns+1:Ns:end); % Échantillonage
        % Decision
        dec_re = sign(real(z_eche));
        dec_ime = sign(imag(z_eche));
        demappinge = (dec_re+1 + j*(dec_ime+1))/2;
        bitse = zeros(1, length(Donneese));
        bitse(1:2:end) = real(demappinge);
        bitse(2:2:end) = imag(demappinge);
        teste = bitse - Donnees;
        TEB_bruite(i) = length(find(teste~=0))/length(bits);
    end
    tmpe = tmpe + TEB_bruite;
end
TEB_bruite = tmpe/precision;
figure();
semilogy(eb_N0_db, TEB_th, 'DisplayName', 'TEB théorique');hold on
semilogy(eb_N0_db, TEB_bruite, 'DisplayName', 'TEB simulé');
legend;
title('Comparaison TEB');
hold off

figure();
scatterplot(mapping);
title("Les constellations en sortie du mapping");
scatterplot(z_eche);
title("Les constellations en sortie de l'échantillonneur");

figure();
semilogy(eb_N0_db, TEB_bruit, 'DisplayName', 'TEB chaine');hold on
semilogy(eb_N0_db, TEB_bruite, 'DisplayName', 'TEB equivalente');
legend;
title('Comparaison TEB chaine et equivalente');
hold off
```

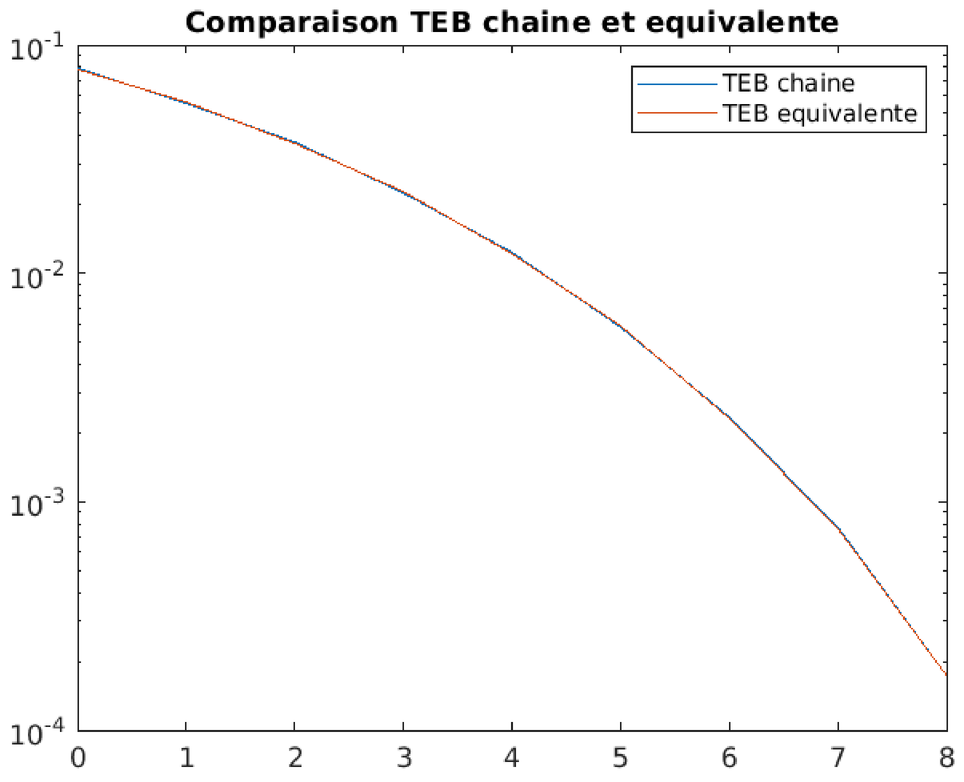

Les constellations en sortie de l'échantillonneur
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


DSP de l'envellope complexe
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5000
0
5000


Comparaison TEB
TEB théorique
TEB simulé
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
7
8


Les constellations en sortie du mapping
Quadrature
In-Phase
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Les constellations en sortie de l'échantillonneur
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2



Comparaison de modulations sur fréquence porteuse

```
h = rcosdesign(0.35,8,Ns); % Filtre en cosinus
% Étude sans bruit

% 4-ASK
% Mapping
mapping_ask = (2*bi2de(reshape(bits, 2, length(bits)/2).') - 3).'; %
 Du binaire au decimal
% Suréchantillonage
Kronr_ask = kron(mapping_ask, [1 zeros(1, Ns-1)]); % Positionnement
 des zéros entre les ak
signalfiltre_ask = filter(h ,1, [Kronr_ask zeros(1,Ns*8)]); % Filtrage
DSP_ASK = pwelch(signalfiltre_ask, [], [], Fe, "centered"); % Densité
 spectrale de puissance
DSP_ASK = DSP_ASK./max(DSP_ASK); % Normaliser la DSP
% Filtrage de réception
z_sb_ask = filter(hr, 1, signalfiltre_ask); % signal filtré par le
 filtre réctangulaire
% Echantillonage
z_ech_sb_ask = z_sb_ask(8*Ns+1:Ns:end); % échantillonage du
 rectangulaire
% Decision
```

```
cond_sb1 = find(z_ech_sb_ask > 2);
cond_sb2 = find(z_ech_sb_ask >= 0 & z_ech_sb_ask <= 2);
cond_sb3 = find(z_ech_sb_ask < 0 & z_ech_sb_ask > -2);
cond_sb4 = find(z_ech_sb_ask <= -2);
dec_sb_ask = zeros(1, length(z_ech_sb_ask));
dec_sb_ask(cond_sb1) = 3;
dec_sb_ask(cond_sb2) = 1;
dec_sb_ask(cond_sb3) = -1;
dec_sb_ask(cond_sb4) = -3;
% Demapping
dem_sb_ask = reshape(de2bi((dec_sb_ask + 3)/2).', 1, length(bits));
% Taux d'erreur binaire
test_sb_ask = dem_sb_ask - bits;
TEB_sb_ask = length(find(test_sb_ask~=0))/length(bits) % Taux d'Erreur
 Binaire

% Q-psk
bits_QPSK = reshape(bits,length(bits)/2,2);
bits_QPSK = bi2de(bits_QPSK)';
symgray = pskmod(bits_QPSK,4,pi/4,'gray');
kronqpsk = kron(symgray, [1 zeros(1, Ns-1)]);
signalfiltre_psk = filter(h ,1, [kronqpsk zeros(1,Ns*8)]);
DSP_QPSK = pwelch(signalfiltre_psk, [], [], Fe, "centered"); % Densité
 spectrale de puissance
DSP_QPSK = DSP_QPSK./max(DSP_QPSK); % Normaliser la DSP
zepsk = filter(hr, 1, signalfiltre_psk);
echq_psk = zepsk(8*Ns+1:Ns:end);
bits_psk = de2bi(pskdemod(echq_psk,4,pi/4,'gray'), 2);
test_psk = bits_psk(:)' - bits;
TEB_sb_Qpsk = length(find(test_psk~=0))/length(bits)

% 8-psk
bits_8PSK = reshape([bits 0 0],length([bits 0 0])/3,3);
bits_8PSK = bi2de(bits_8PSK)';
symgray = pskmod(bits_8PSK,8,pi/8,'gray');
kron8psk = kron(symgray, [1 zeros(1, Ns-1)]);
signalfiltre_8psk = filter(h ,1, [kron8psk zeros(1,Ns*8)]);
DSP_8PSK = pwelch(signalfiltre_8psk, [], [], Fe, "centered"); %
 Densité spectrale de puissance
DSP_8PSK = DSP_8PSK./max(DSP_8PSK); % Normaliser la DSP
zepsk8 = filter(hr, 1, signalfiltre_8psk);
ech8psk = zepsk8(8*Ns+1:Ns:end);
bits_8psk = de2bi(pskdemod(ech8psk,8,pi/8,'gray'), 3);
test_8psk = bits_8psk(:)' - [bits 0 0];
TEB_sb_8psk = length(find(test_8psk~=0))/length(bits)

% 16-qam
bits_QAM = reshape(bits, length(bits)/4, 4);
bits_QAM = bi2de(bits_QAM)';
modqam = qammod(bits_QAM,16,'gray');
kronqam = kron(modqam, [1 zeros(1, Ns-1)]);
signalfiltre_qam = filter(h ,1, [kronqam zeros(1,Ns*8)]);
DSP_QAM = pwelch(signalfiltre_qam, [], [], Fe, "centered"); % Densité
 spectrale de puissance
```

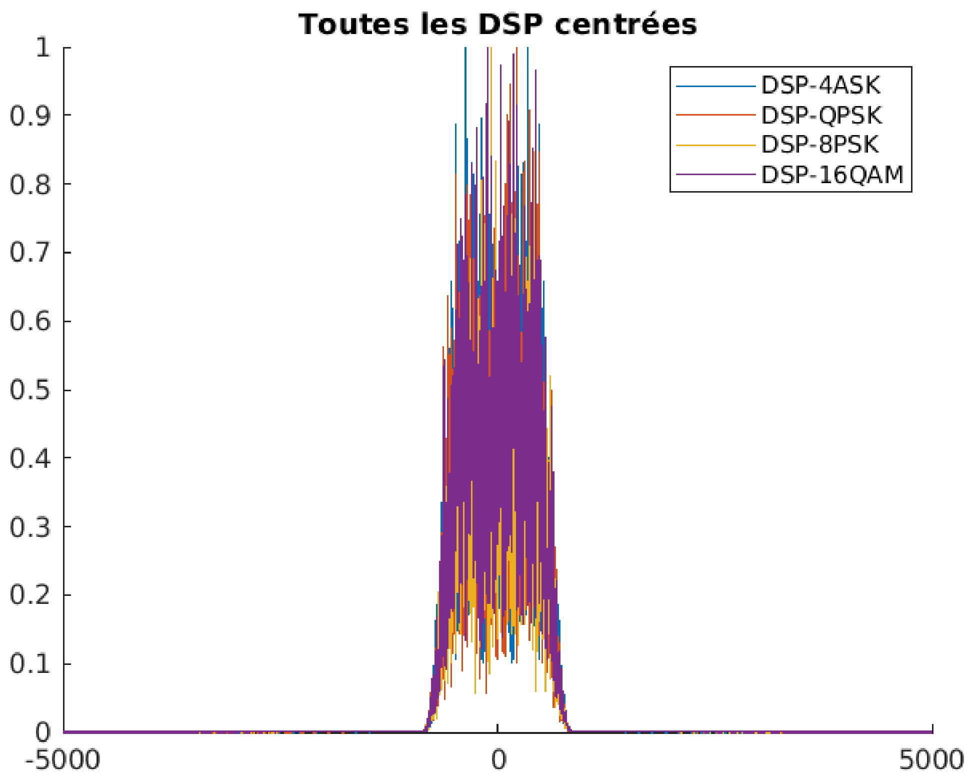
```
DSP_QAM = DSP_QAM./max(DSP_QAM); % Normaliser la DSP
zeqam = filter(hr, 1, signalfiltre_qam);
echqam = zeqam(8*Ns+1:Ns:end);
bits_qam = de2bi(qamdemod(echqam,16,'gray'), 4);
test_qam =  bits_qam(:)' - bits;
TEB_sb_qam = length(find(test_qam~=0))/length(bits)

% Toutes les DSP
figure();
title("Toutes les DSP centrées");
hold on
plot(frequence, DSP_ASK, "DisplayName", "DSP-4ASK");
plot(frequence, DSP_QPSK, "DisplayName", "DSP-QPSK");
plot(frequence, DSP_8PSK, "DisplayName", "DSP-8PSK");
plot(frequence, DSP_QAM, "DisplayName", "DSP-16QAM");
hold off
legend;

% Étude avec bruit

% 4-ASK
tmpask = 0;
for k = 1 : precision
    Donneesask = randi([0 1], 1, 10000);
    for i = 1 : length(eb_N0_db)
        mapping_ask = (2*bi2de(reshape(Donneesask, 2,
 length(Donneesask)/2).') - 3).'; % Du binaire au decimal
        Kronr_ask = kron(mapping_ask, [1 zeros(1, Ns-1)]); %
 Positionnement des zéros entre les ak
        signalfiltre_ask = filter(h ,1, [Kronr_ask zeros(1,Ns*8)]); %
 Filtrage
        Pxask = mean(abs(signalfiltre_ask).^2);
        sigma2 = Pxask*Ns/(4*10^(eb_N0_db(i)/10));
        bruitask = sqrt(sigma2)*randn(1, length(signalfiltre_ask));
        Z_ask = signalfiltre_ask + bruitask;
        z_ask = filter(hr, 1, Z_ask); % Signal filtré
        z_ech_ask = z_ask(8*Ns+1:Ns:end); % Échantillonage
        % Decision
        cond1 = find(z_ech_ask > 2);
        cond2 = find(z_ech_ask >= 0 & z_ech_ask <= 2);
        cond3 = find(z_ech_ask < 0 & z_ech_ask > -2);
        cond4 = find(z_ech_ask <= -2);
        dec_ask(cond1) = 3;
        dec_ask(cond2) = 1;
        dec_ask(cond3) = -1;
        dec_ask(cond4) = -3;
        dem_ask = reshape(de2bi((dec_ask + 3)/2).', 1,
 length(Donneesask));
        test_ask = dem_ask - Donneesask;
        TEB_bruit_ask(i) = length(find(test_ask~=0))/
length(mapping_ask);
    end
    tmpask = tmpask + TEB_bruit_ask;
end
```

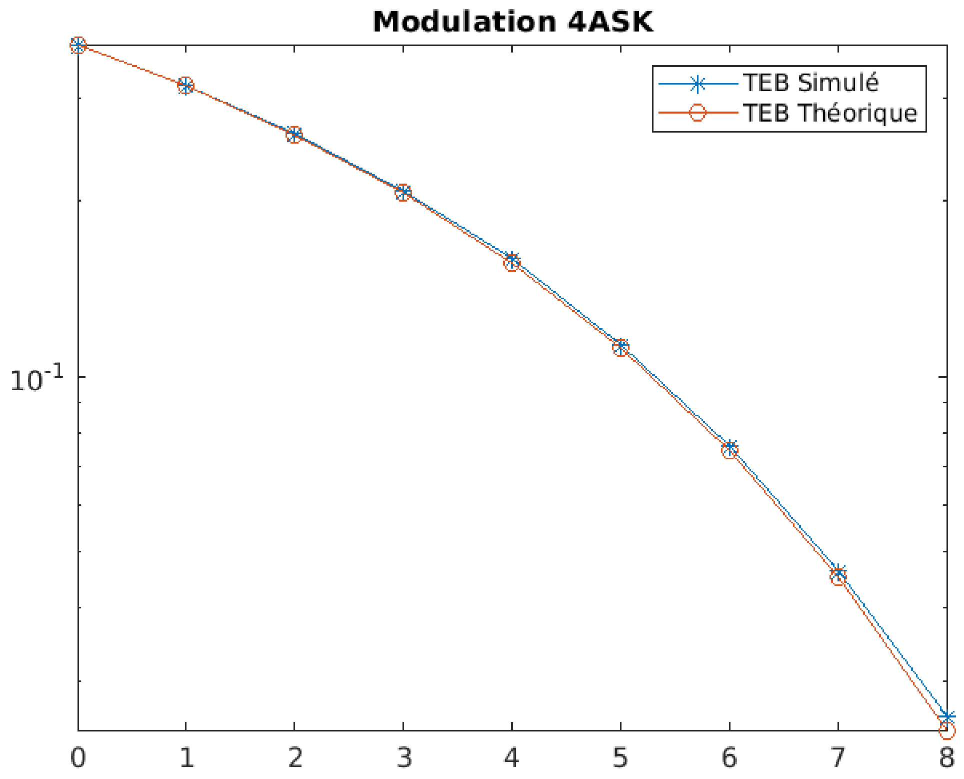
```
figure();
TEB_bruit_ask = tmpask/precision;
figure();
semilogy(eb_N0_db,TEB_bruit_ask,'*-');
hold on;
TEB_4ASK_Theorique = 2*qfunc(sqrt((12/15)*10.^(eb_N0_db/10)));
semilogy(eb_N0_db,TEB_4ASK_Theorique,'o-');
legend("TEB Simulé","TEB Théorique");
title("Modulation 4ASK");
hold off

figure();
scatterplot(mapping_ask);
title("Les constellations en sortie du mapping 4-ASK");
scatterplot(z_ech_ask);
title("Les constellations en sortie de l'échantillonneur 4-ASK");

% Q-PSK
tmpsk = zeros(1,length(eb_N0_db));
for k = 1:precision
    Donneespsk = randi([0 1], 1, 10000);
    for i = 1 : length(eb_N0_db)
        bits_QPSK = reshape(Donneespsk,length(bits)/2,2);
        bits_QPSK = bi2de(bits_QPSK)';
        symgray = pskmod(bits_QPSK,4,pi/4,'gray');
        kronqpsk = kron(symgray, [1 zeros(1, Ns-1)]);
        signalfiltre_psk = filter(h ,1, [kronqpsk zeros(1,Ns*8)]);
        Pxpsk = mean(abs(signalfiltre_psk).^2);
        sigma2 = Pxpsk*Ns/(2*log2(M)*10^(eb_N0_db(i)/10));
        bruitrpsk = sqrt(sigma2)*randn(1,
 length(real(signalfiltre_psk)));
        bruitipsk = sqrt(sigma2)*randn(1,
 length(imag(signalfiltre_psk)));
        bruitpsk = bruitrpsk + j*bruitipsk;
        Zepsk = signalfiltre_psk + bruitpsk;
        zepsk = filter(hr, 1, Zepsk);
        echq_psk = zepsk(8*Ns+1:Ns:end);
        bits_psk = de2bi(pskdemod(echq_psk,4,pi/4,'gray'), 2);
        test_psk = bits_psk(:)' - Donneespsk;
        TEB_bruit_psk(i) = length(find(test_psk~=0))/length(bits);
    end
    tmpsk = tmpsk + TEB_bruit_psk;
end
figure();
TEB_bruit_psk = tmpsk/precision;
semilogy(eb_N0_db,TEB_bruit_psk,'*-');
hold on;
TEB_QPSK_Theorique = qfunc(sqrt(2*10.^(eb_N0_db/10)));
semilogy(eb_N0_db,TEB_QPSK_Theorique,'o-');
legend("TEB Simulé","TEB Théorique");
title("Modulation QPSK");
hold off

figure();
```

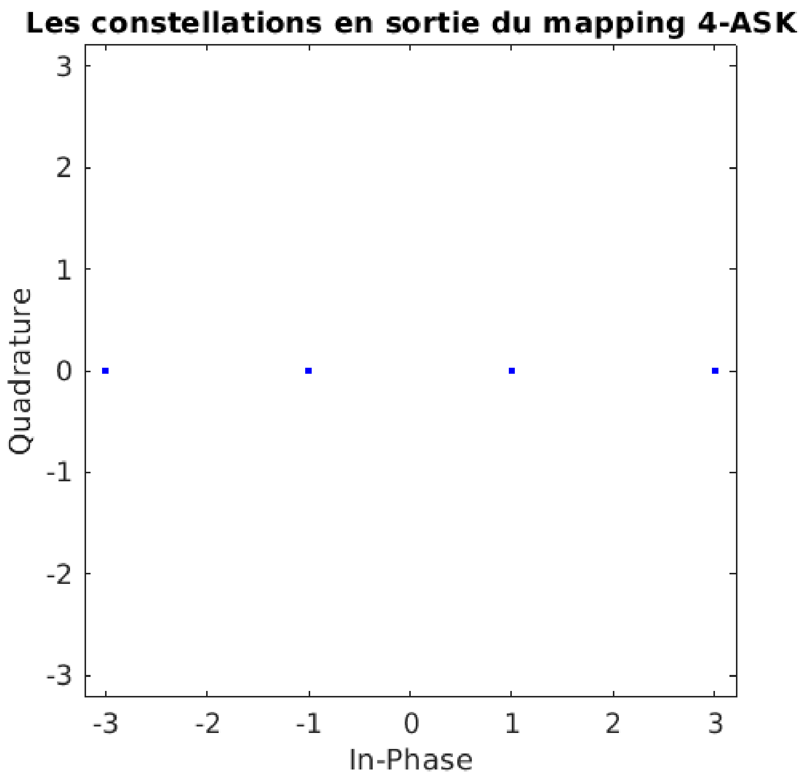
```
scatterplot(symgray);
title("Les constellations en sortie du mapping Q-PSK");
scatterplot(echq_psk);
title("Les constellations en sortie de l'échantillonneur Q-PSK");

% 8-PSK
tmpsk8 = zeros(1,length(eb_N0_db));
for k = 1:precision
    Donneespsk = randi([0 1], 1, 10000);
    for i = 1 : length(eb_N0_db)
        bits_8PSK = reshape([Donneespsk 0 0],length([bits 0 0])/3,3);
        bits_8PSK = bi2de(bits_8PSK)';
        symgray = pskmod(bits_8PSK,8,pi/8,'gray');
        kron8psk = kron(symgray, [1 zeros(1, Ns-1)]);
        signalfiltre_8psk = filter(h ,1, [kron8psk zeros(1,Ns*8)]);
        Pxpsk = mean(abs(signalfiltre_8psk).^2);
        sigma2 = Pxpsk*Ns/(2*log2(8)*10^(eb_N0_db(i)/10));
        bruitrpsk = sqrt(sigma2)*randn(1,
 length(real(signalfiltre_8psk)));
        bruitipsk = sqrt(sigma2)*randn(1,
 length(imag(signalfiltre_8psk)));
        bruitpsk = bruitrpsk + j*bruitipsk;
        Zepsk8 = signalfiltre_8psk + bruitpsk;
        zepsk8 = filter(hr, 1, Zepsk8);
        ech8psk = zepsk8(8*Ns+1:Ns:end);
        bits_8psk = de2bi(pskdemod(ech8psk,8,pi/8,'gray'), 3);
        test_8psk = bits_8psk(:)' - [Donneespsk 0 0];
        TEB_bruit_psk8(i) = length(find(test_8psk~=0))/length(bits);
    end
    tmpsk8 = tmpsk8 + TEB_bruit_psk8;
end
figure();
TEB_bruit_psk8 = tmpsk8/precision;
semilogy(eb_N0_db,TEB_bruit_psk8,'*-');
hold on;
TEB_8PSK_Theorique = 2/3*qfunc(sqrt(2*sin(pi/8)*10.^(eb_N0_db/10)));
semilogy(eb_N0_db,TEB_8PSK_Theorique,'o-');
legend("TEB Simulé","TEB Théorique");
title("Modulation 8PSK");
hold off

figure();
scatterplot(symgray);
title("Les constellations en sortie du mapping 8-PSK");
scatterplot(ech8psk);
title("Les constellations en sortie de l'échantillonneur 8-PSK");

% 16-QAM
tmpqam = zeros(1,length(eb_N0_db));
for k = 1:precision
    Donneesqam = randi([0 1], 1, 10000);
    for i = 1 : length(eb_N0_db)
        bits_QAM = reshape(Donneesqam,length(bits)/4,4);
        bits_QAM = bi2de(bits_QAM)';
        modqam = qammod(bits_QAM,16,'gray');
        kronqam = kron(modqam, [1 zeros(1, Ns-1)]);
        signalfiltre_qam = filter(h ,1, [kronqam zeros(1,Ns*8)]);
        Pxqam = mean(abs(signalfiltre_qam).^2);
        sigma2 = Pxqam*Ns/(2*log2(16)*10^(eb_N0_db(i)/10));
        bruitrqam = sqrt(sigma2)*randn(1,
 length(real(signalfiltre_qam)));
        bruitiqam = sqrt(sigma2)*randn(1,
 length(imag(signalfiltre_qam)));
        bruitqam = bruitrqam + j*bruitiqam;
        Zeqam = signalfiltre_qam + bruitqam;
        zeqam = filter(hr, 1, Zeqam);
        echqam = zeqam(8*Ns+1:Ns:end);
        bits_qam = de2bi(qamdemod(echqam,16,'gray'), 4);
        test_qam =  bits_qam(:)' - Donneesqam;
        TEB_bruit_qam(i) = length(find(test_qam~=0))/length(bits);
    end
    tmpqam = tmpqam + TEB_bruit_qam;
end
figure();
TEB_bruit_qam = tmpqam/precision;
semilogy(eb_N0_db,TEB_bruit_qam,'*-');
hold on;
TEB_QAM_Theorique = (3/4)*qfunc(sqrt((12/15)*10.^(eb_N0_db/10)));
semilogy(eb_N0_db,TEB_QAM_Theorique,'o-');
legend("TEB Simulé","TEB Théorique");
title("Modulation QAM");
hold off

figure();
scatterplot(modqam);
title("Les constellations en sortie du mapping 16-QAM");
scatterplot(echqam);
title("Les constellations en sortie de l'échantillonneur 16-QAM");

figure();
semilogy(eb_N0_db, TEB_bruit_psk, 'DisplayName', 'TEB simulé Q-
PSK');hold on
semilogy(eb_N0_db, TEB_bruit_qam, 'DisplayName', 'TEB simulé 16-QAM');
semilogy(eb_N0_db, TEB_bruit_psk8, 'DisplayName', 'TEB simulé 8-PSK');
semilogy(eb_N0_db, TEB_bruit_ask, 'DisplayName', 'TEB simulé 4-ASK')
legend;
title('Comparaison TEB');
hold off
```

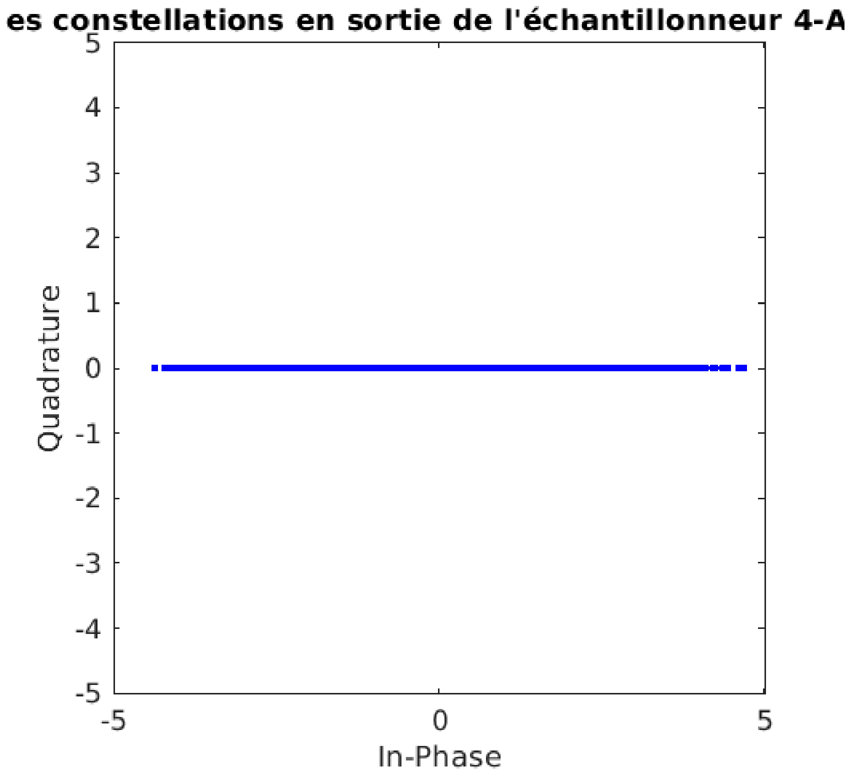
```
TEB_sb_ask =

     0


TEB_sb_Qpsk =

     0
```

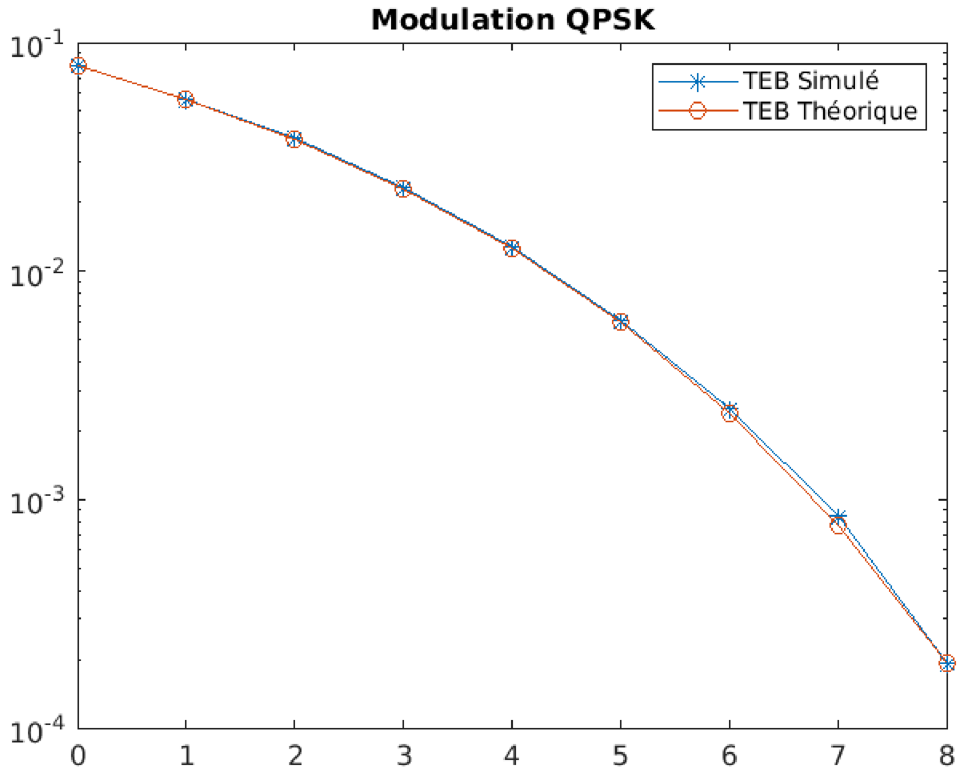
```
TEB_sb_8psk =

     0


TEB_sb_qam =

     0
```

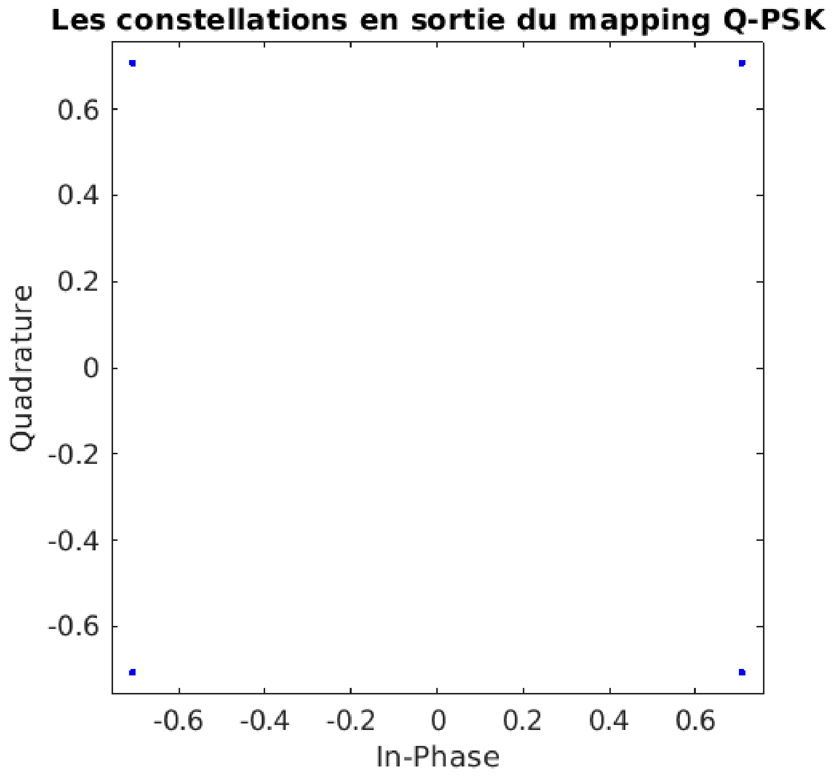


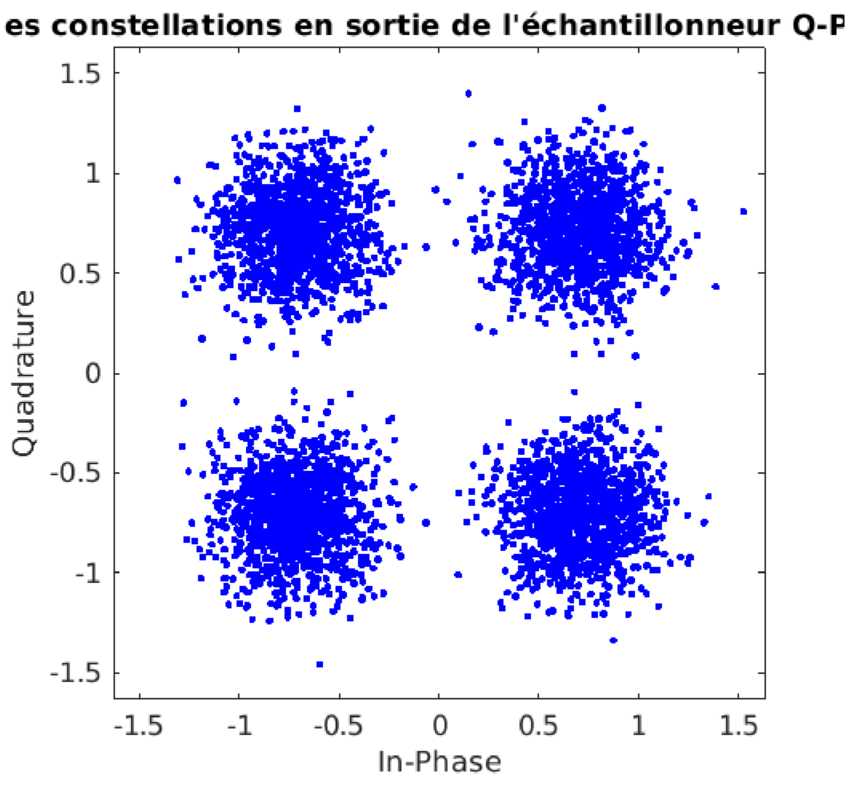

Modulation 4ASK
TEB Simulé
TEB Théorique
10^-1
0
1
2
3
4
5
6
7
8

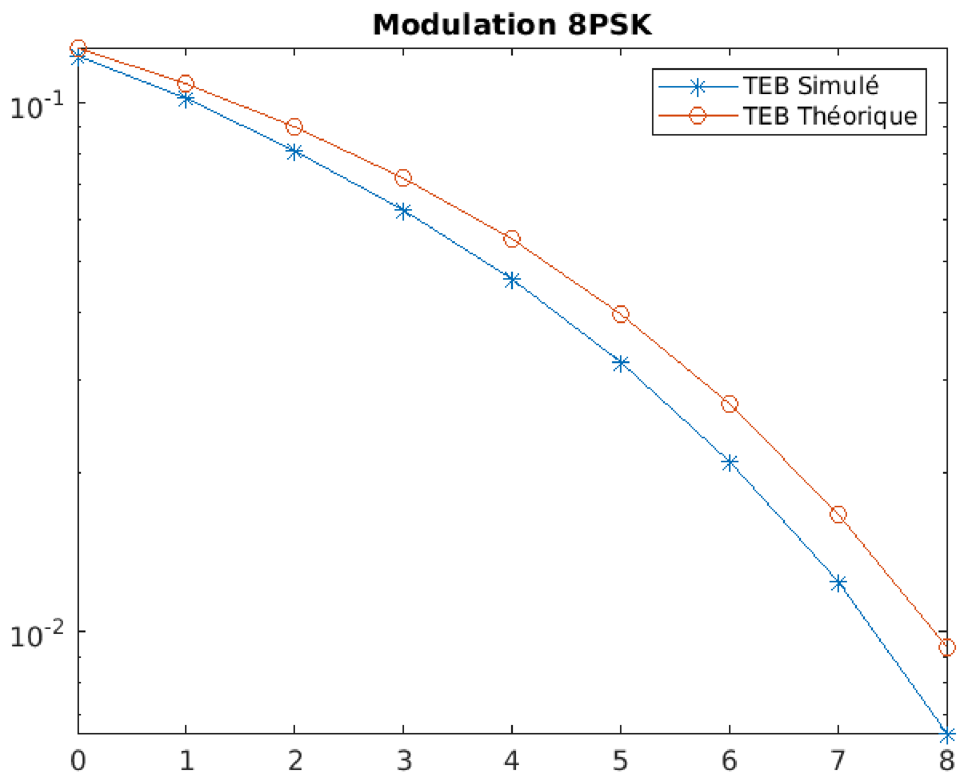

Les constellations en sortie du mapping 4-ASK
Quadrature
In-Phase
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

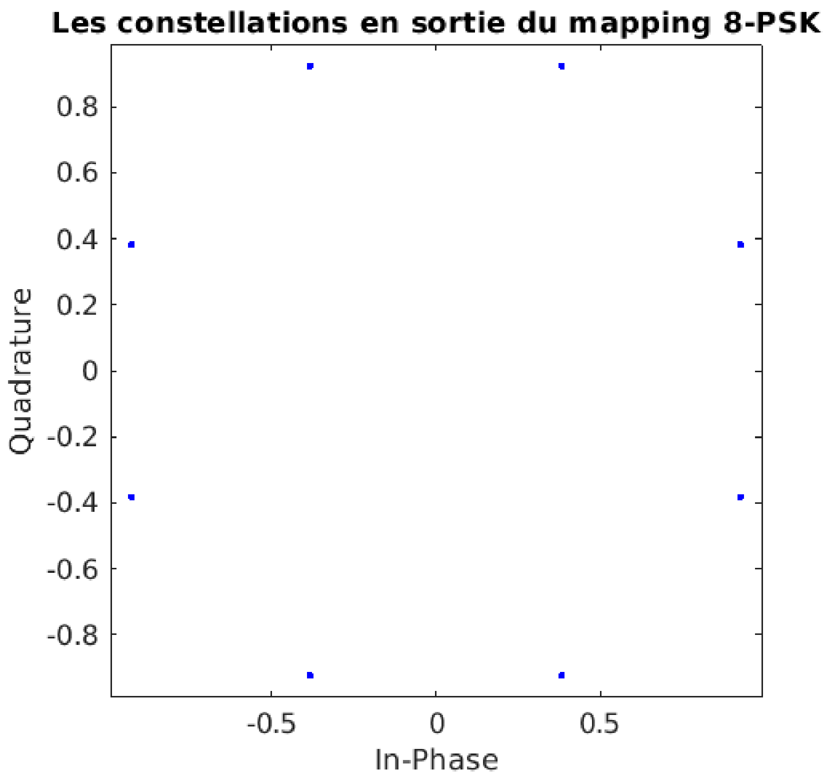

es constellations en sortie de l'échantillonneur 4-A
Quadrature
In-Phase
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
0
5

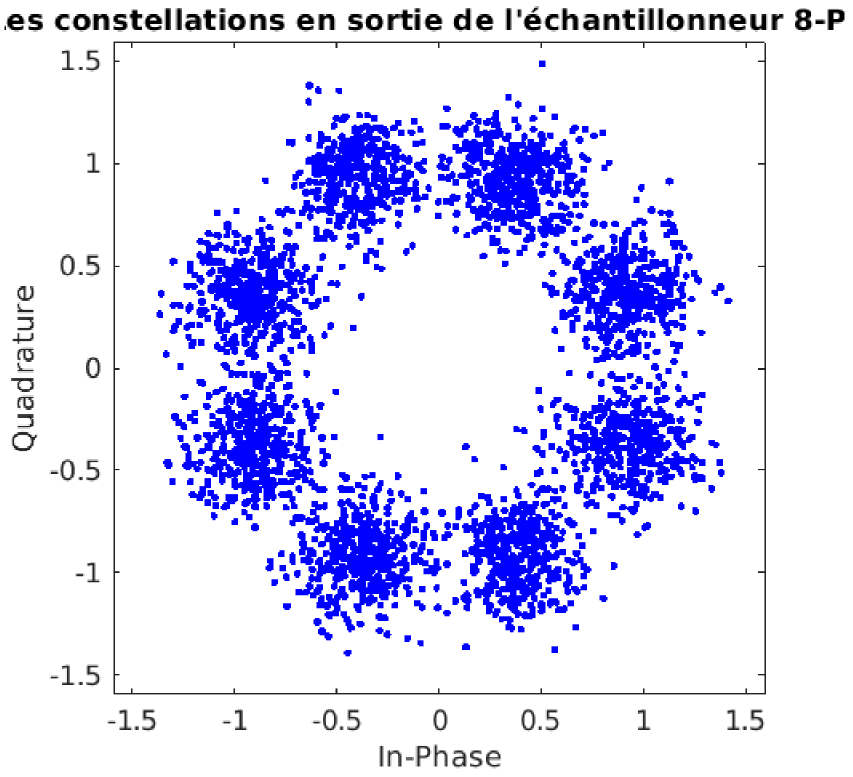

Modulation QPSK
TEB Simulé
TEB Théorique
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
7
8


Les constellations en sortie du mapping Q-PSK
Quadrature
In-Phase
0.6
0.4
0.2
0
-0.2
-0.4
-0.6

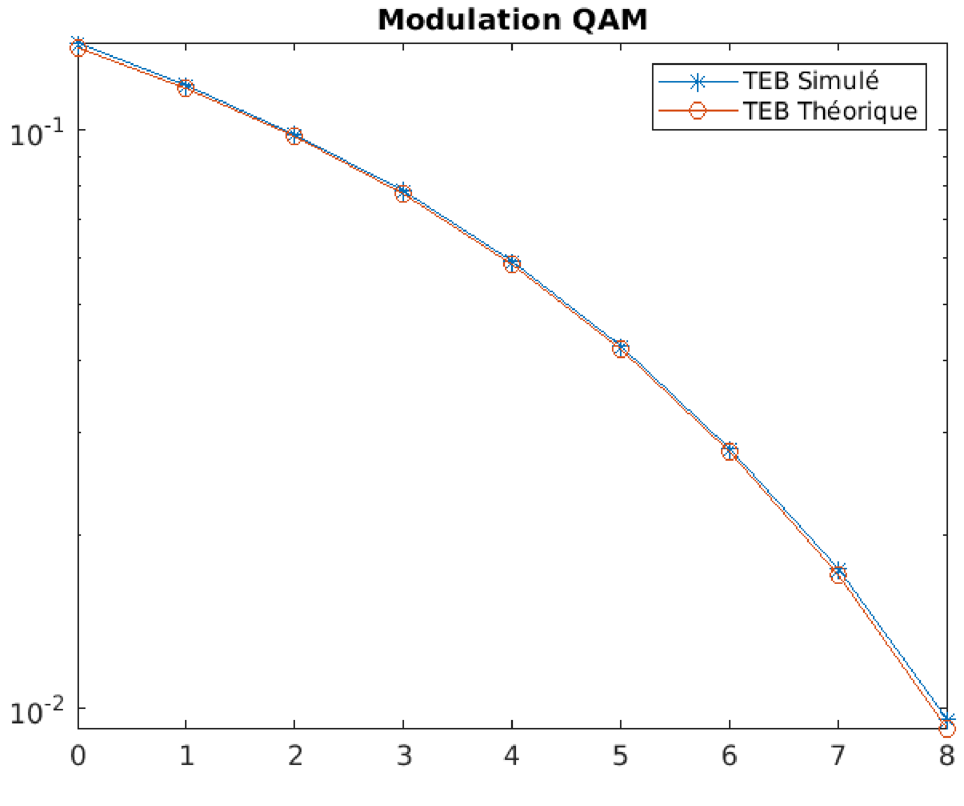

es constellations en sortie de l'échantillonneur Q-P
Quadrature
In-Phase


Modulation 8PSK
TEB Simulé
TEB Théorique

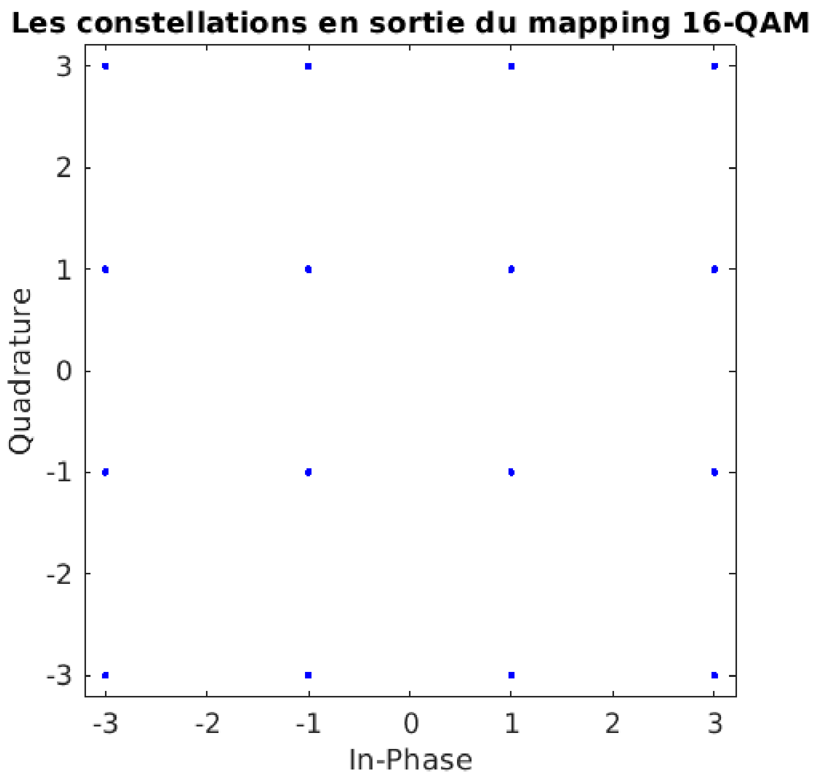

Les constellations en sortie du mapping 8-PSK
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
Quadrature
-0.5
0
0.5
In-Phase

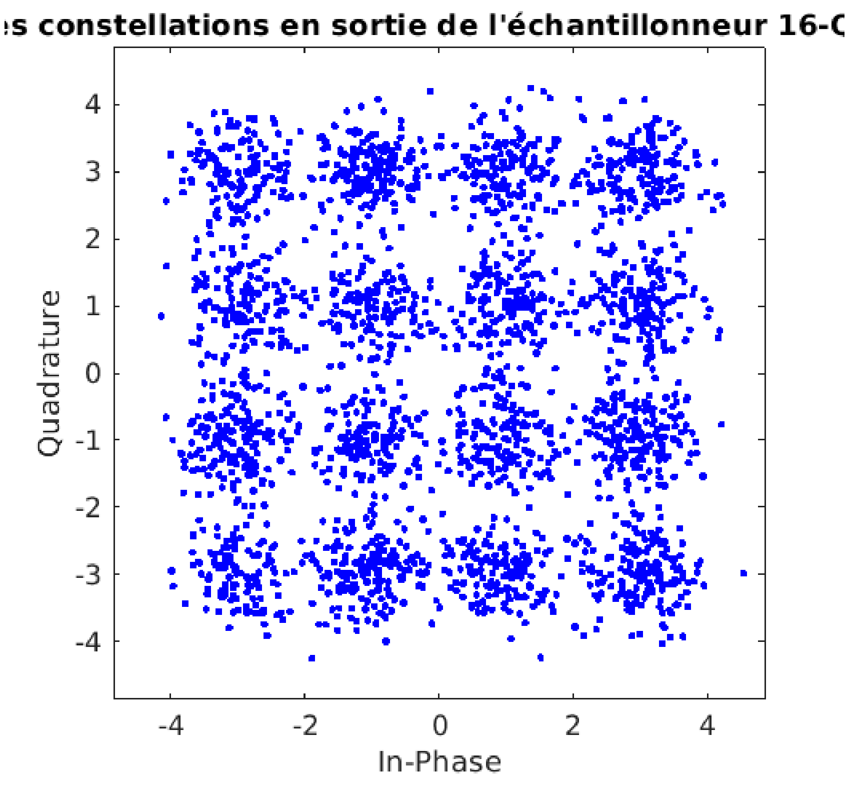


es constellations en sortie de l'échantillonneur 8-P
Quadrature
In-Phase
1.5
1
0.5
0
-0.5
-1
-1.5

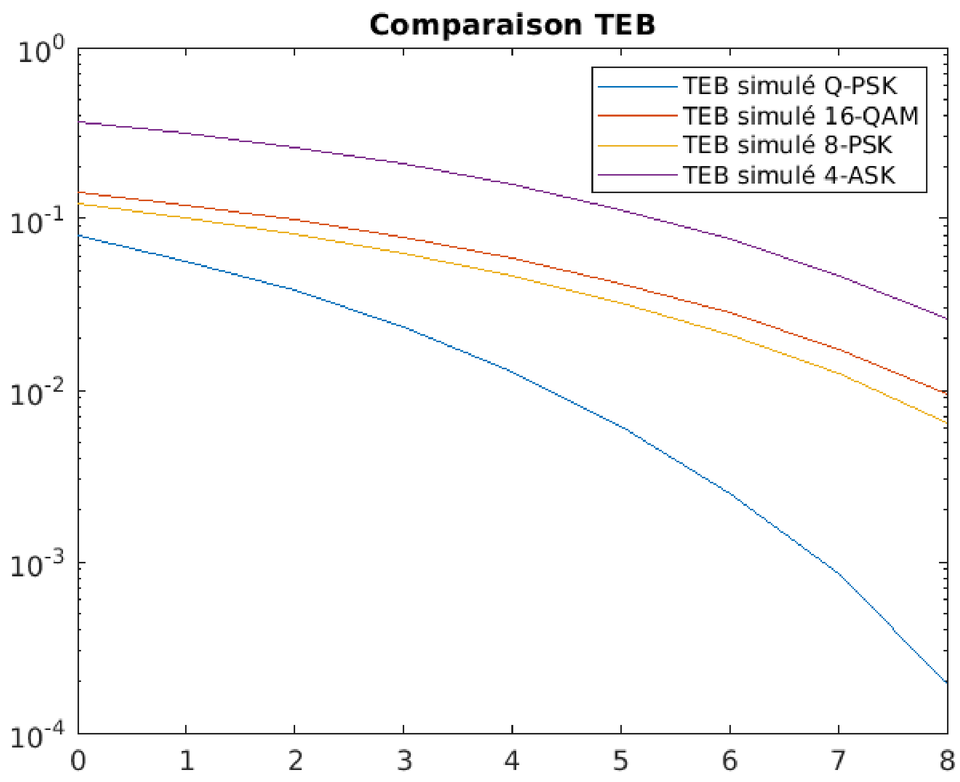


Modulation QAM
TEB Simulé
TEB Théorique
10^{-1}
10^{-2}
0
1
2
3
4
5
6
7
8


Les constellations en sortie du mapping 16-QAM
Quadrature
In-Phase
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3



ːs constellations en sortie de l'échantillonneur 16-C
Quadrature
In-Phase



Comparaison TEB
TEB simulé Q-PSK
TEB simulé 16-QAM
TEB simulé 8-PSK
TEB simulé 4-ASK

Published with MATLAB® R2020a