Table of Contents

Projet Partie II A23 ELE6701A

Bouh Abdillahi

Matricule : 1940646

Github : https://github.com/konoDioDA253/ELE6701A

```
clear all;
clc;
H=[1 0 0 0 0 0 0 0 0 0;
    -0.4 1 0 0 0 0 0 0 0 0;
    0 -0.4 1 0 0 0 0 0 0 0;
    0  0 -0.4 1 0 0 0 0 0 0;
    0 0  0 -0.4 1 0 0 0 0 0;
    0 0 0  0 -0.4 1 0 0 0 0;
    0 0 0 0  0 -0.4 1 0 0 0;
    0 0 0 0 0  0 -0.4 1 0 0;
    0 0 0 0 0 0  0 -0.4 1 0;
    0 0 0 0 0 0 0  0 -0.4 1];
```

G?n?ration des 1024 vecteurs hypoth?ses

```
vecteurs_cell = cell(1, 1024);

for i = 1:1024
    % Convertir l'indice en binaire sur 10 bits
    binaire = dec2bin(i-1, 10) - '0'; % Convertir en vecteur binaire

    % Remplacer les '0' par '-1' dans la repr?sentation binaire
    binaire(binaire==0) = -1;

    % Stocker le vecteur dans la cellule
    vecteurs_cell{i} = binaire';
end

% Convertir la cellule en une matrice de taille 10x1024
matrice_vecteurs_hypothese_s = cell2mat(vecteurs_cell);
```

G?n?ration du bruit n

Variance d?sir?e

```
variance_desiree = (0.15);

% Nombre de vecteurs ? g?n?rer
nombre_paquets = 10000; %% VALEUR A MODIFIER
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% G?n?ration des 1024 vecteurs de bruit gaussien blanc avec moyenne
 nulle
vecteurs_bruit_gaussien = sqrt(variance_desiree) *
 randn(taille_vecteur, nombre_vecteurs);
```

G?n?ration de paquets s_i ? envoyer

Nombre de vecteurs al?atoires ? g?n?rer

```
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% G?n?ration des vecteurs al?atoires
vecteurs_aleatoires_envoyes = randi([1, 2], taille_vecteur,
 nombre_vecteurs);
vecteurs_aleatoires_envoyes(vecteurs_aleatoires_envoyes == 2) = -1;
```

Vecteur y re?u

Matrice des vecteurs y

```
y=vecteurs_aleatoires_envoyes + vecteurs_bruit_gaussien;
```

Parametres pour calcul egalisteur MMSE

```
Rs = eye(3);
Rn = variance_desiree*Rs;
Delta=[0 1 2];

H=[1 -0.4 0 0;
    0 1 -0.4 0;
    0 0 1 -0.4];
eDelta1 = [1;0;0;0];
eDelta2 = [0;1;0;0];
eDelta3 = [0;0;1;0];
Ry = Rs*H*(H') + Rn;
```

Egalisateur MMSE

```
w_Delta1 = (eDelta1')*(H')*Ry^-1 % coefficients pour Delta = 0
```

```
w_Delta2 = (eDelta2')*(H')*Ry^-1 % coefficients pour Delta = 1
w_Delta3 = (eDelta3')*(H')*Ry^-1 % coefficients pour Delta = 2

MMSE_Delta1_theorique = 1 - w_Delta1*Ry*(w_Delta1')
MMSE_Delta2_theorique = 1 - w_Delta2*Ry*(w_Delta2')
MMSE_Delta3_theorique = 1 - w_Delta3*Ry*(w_Delta3')
```

```
w_Delta1 =

    0.8508    0.2865    0.0875


w_Delta2 =

   -0.0538    0.8237    0.2515


w_Delta3 =

   -0.0271   -0.0888    0.7362


MMSE_Delta1_theorique =

    0.1492


MMSE_Delta2_theorique =

    0.1547


MMSE_Delta3_theorique =

    0.2282
```

Decision MMSE

s_chapeau_iDelta1 = zeros(10,nombre_paquets);

```
s_chapeau_iDelta2 =  zeros(10,nombre_paquets);
% s_chapeau_iDelta3 =  zeros(10,nombre_paquets);

% vecteur_guess_s_Delta1 = zeros(10,nombre_paquets);
vecteur_guess_s_Delta2 = zeros(10,nombre_paquets);
% vecteur_guess_s_Delta3 = zeros(10,nombre_paquets);
error_count_symbol_bii = 0;
error_count_vector_bii = 0;
error_count_symbol_biii = 0;
error_count_vector_biii = 0;
for i=1:1:nombre_paquets
%     s_chapeau_iDelta1(:,i) = w1*[y(1,i); y(); y()];
```

```
    s_chapeau_iDelta2(:,i) = [w_Delta2*[y(2,i); y(1,i); 0];
                                                    w_Delta2*[y(3,i); y(2,i);
 y(1,i)];
                                                     w_Delta2*[y(4,i); y(3,i);
 y(2,i)];
                                                     w_Delta2*[y(5,i); y(4,i);
 y(3,i)];
                                                     w_Delta2*[y(6,i); y(5,i);
 y(4,i)];
                                                     w_Delta2*[y(7,i); y(6,i);
 y(5,i)];
                                                     w_Delta2*[y(8,i); y(7,i);
 y(6,i)];
                                                     w_Delta2*[y(9,i); y(8,i);
 y(7,i)];
                                                     w_Delta2*[y(10,i);
 y(9,i); y(8,i)];
                                                     w_Delta2*[0; y(10,i);
 y(9,i)];
                                                    ];

%     s_chapeau_iDelta3(:,i) = w3*y(:,i);

    for j=1:1:10
%         if s_chapeau_iDelta1(j,i) < 0
%             vecteur_guess_s_Delta1(j,i) = -1;
%         else
%             vecteur_guess_s_Delta1(j,i) = 1;
%         end
%
        if s_chapeau_iDelta2(j,i) < 0
            vecteur_guess_s_Delta2(j,i) = -1;
        else
            vecteur_guess_s_Delta2(j,i) = 1;
        end

%         if s_chapeau_iDelta3(j,i) < 0
%             vecteur_guess_s_Delta3(j,i) = -1;
%         else
%             vecteur_guess_s_Delta3(j,i) = 1;
%         end
    end

    % question bii
    comp = isequal(vecteurs_aleatoires_envoyes(:,i),
 vecteur_guess_s_Delta2(:,i));
    difference_per_symbol_bii = nnz(vecteur_guess_s_Delta2(:,i) ~=
 vecteurs_aleatoires_envoyes(:,i));
    error_count_symbol_bii = error_count_symbol_bii +
 difference_per_symbol_bii;
    if comp
        test = 0;
    else
```

```
        error_count_vector_bii = error_count_vector_bii+1;
%         disp("error!");
    end

    % question biii
    % On considere que y passe par le module de decision avant de le
    % comparer aux vecteurs envoyes
    y_post_decision_vecteur = y(:,i);
    y_post_decision_vecteur(y_post_decision_vecteur > 0) = 1;   % ?l?
ments positifs deviennent 1
    y_post_decision_vecteur(y_post_decision_vecteur < 0) = -1;  % ?l?
ments n?gatifs deviennent -1
    difference_per_symbol_biii = nnz(y_post_decision_vecteur ~=
 vecteurs_aleatoires_envoyes(:,i));
    error_count_symbol_biii = error_count_symbol_biii +
 difference_per_symbol_biii;
end

taux_erreur_par_paquet_de_10_symboles_bii = error_count_vector_bii/
nombre_paquets
taux_erreur_par_symbole_bii = error_count_symbol_bii/
(nombre_paquets*10)

diff_carree = (s_chapeau_iDelta2 - vecteurs_aleatoires_envoyes).^2;
mmse_question_bi = mean(diff_carree(:))

% Pour le mmse biii on considere que y passe directement par le module
 de
% d?cision (sans egalisateur)
% Changement des valeurs en fonction du signe
y_post_decision = y;
y_post_decision(y_post_decision > 0) = 1;   % ?l?ments positifs
 deviennent 1
y_post_decision(y_post_decision < 0) = -1;  % ?l?ments n?gatifs
 deviennent -1
diff_carree = (y_post_decision - vecteurs_aleatoires_envoyes).^2;
mmse_question_biii = mean(diff_carree(:))

taux_erreur_par_symbole_biii = error_count_symbol_biii/
(nombre_paquets*10)


taux_erreur_par_paquet_de_10_symboles_bii =

    0.2012


taux_erreur_par_symbole_bii =

    0.0218


mmse_question_bi =
```

```
    0.2015
```

```
mmse_question_biii =
```

```
    0.0213
```

```
taux_erreur_par_symbole_biii =
```

```
    0.0053
```

Commentaire bii

On remarque que l'erreur est de l'ordre de 2% pour chaque symbole. Cela repr?sente une augmentation significative par rapport ? la question I-d o? nous ?tions de l'ordre de 0.4%

Commentaire biii

On remarque que le mmse en II-biii est plus faible qu'en II-bi. Par ailleurs on voit que le taux d'erreur par symbole est de 0.15% pour II-bii et est environ 0.42%% en II-biii. L'egalisateur fonctionne donc bel et bien car il donne un taux d'erreur et un mmse plus bas.

Published with MATLAB® R2019b