# Table of Contents



# Projet Partie III A23 ELE6701A

Bouh Abdillahi

Matricule : 1940646

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

```
clear all;
clc;
% Choix du systeme d'exploitation
is_unix = 0; % 0 si votre systeme est windows; 1 sinon (Mac,Ubuntu)
if is_unix
    % Unix
    path = "fig/";
else
    % Windows
    path = "fig\";
end
% Creation du repertoire des figures
mkdir('fig');
% Creation des noms des figures
nom_image = ["Entrainement_IIIa"];
```

```
Warning: Directory already exists.
```

# Generation du bruit n

Variance desiree

```
variance_desiree = (0.15);

% Nombre de vecteurs a generer
nombre_paquets = 10000; %% VALEUR A MODIFIER
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% Generation des vecteurs de bruit gaussien blanc avec moyenne nulle
vecteurs_bruit_gaussien = sqrt(variance_desiree) *
 randn(taille_vecteur, nombre_vecteurs);
```

# Generation de paquets s_i a envoyer

Nombre de vecteurs aleatoires a generer

```
nombre_vecteurs = nombre_paquets;
taille_vecteur = 10;

% Generation des vecteurs aleatoires
vecteurs_aleatoires_envoyes = randi([1, 2], taille_vecteur,
 nombre_vecteurs);
vecteurs_aleatoires_envoyes(vecteurs_aleatoires_envoyes == 2) = -1;
```

# Vecteur y recu

Matrice des vecteurs y

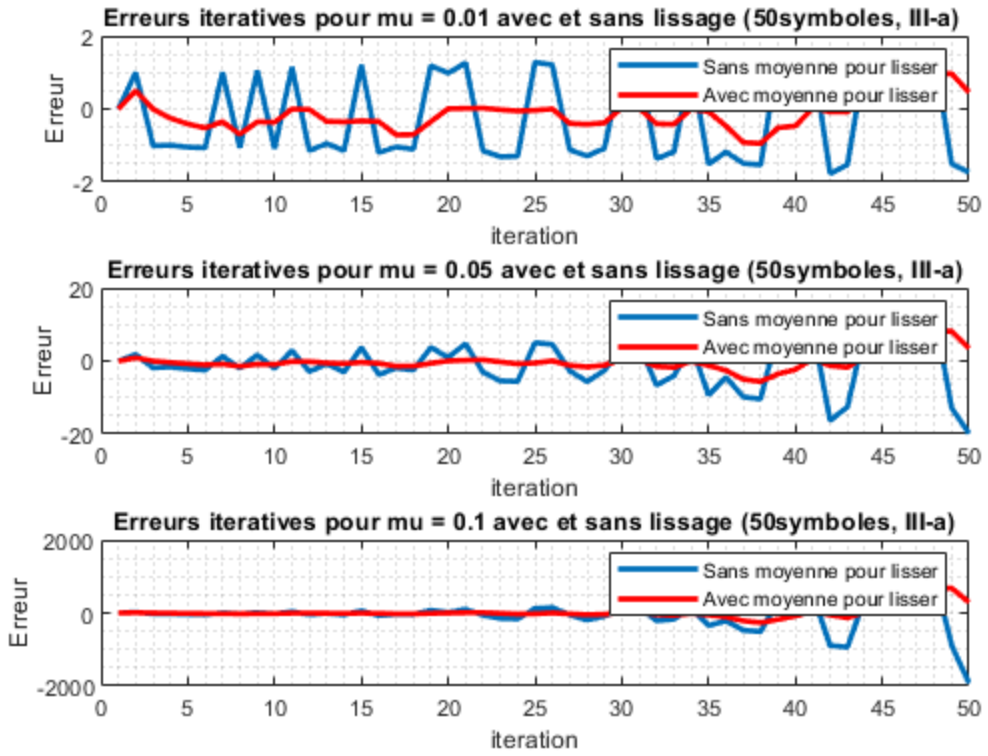
```
y=vecteurs_aleatoires_envoyes + vecteurs_bruit_gaussien;
```

# Egalisateur LMS (entrainement avec 50 symboles)

```
dataset_size = 50; %% INPUT, A MODIFIER
w=zeros(3,1);
y_list = y(:);
vecteurs_aleatoires_envoyes_list = vecteurs_aleatoires_envoyes(:);
erreur_iterative_list = 233423411*ones(dataset_size,3);
compteur_mu=1;
% Initialisation du vecteur pour stocker les moyennes
moyennes_erreur_iterative_list = zeros(dataset_size, 3);
s_chapeau = 34342342*ones(dataset_size,3);
s_guess = 34342342*ones(dataset_size,3);
s_envoye = vecteurs_aleatoires_envoyes_list(1:dataset_size);
for mu=[0.01 0.05 0.1]
    for i=1:1:dataset_size
        if i == 1
            s_chapeau(i,compteur_mu) = [y_list(i), 0, 0]*w;
            erreur_iterative = 0 - [y_list(i), 0, 0]*w;
            w=w-mu*[y_list(i); 0; 0]*(erreur_iterative);
        end
        if i == 2
            s_chapeau(i,compteur_mu) = [y_list(i), y_list(i-1), 0]*w;
            erreur_iterative = vecteurs_aleatoires_envoyes_list(i-1) -
 [y_list(i), y_list(i-1), 0]*w;
            w=w-mu*[y_list(i); y_list(i-1); 0]*(erreur_iterative);
        end
        if i > 2
            s_chapeau(i,compteur_mu) = [y_list(i), y_list(i-1),
 y_list(i-2)]*w;
            erreur_iterative = vecteurs_aleatoires_envoyes_list(i-1) -
 [y_list(i), y_list(i-1), y_list(i-2)]*w;
```

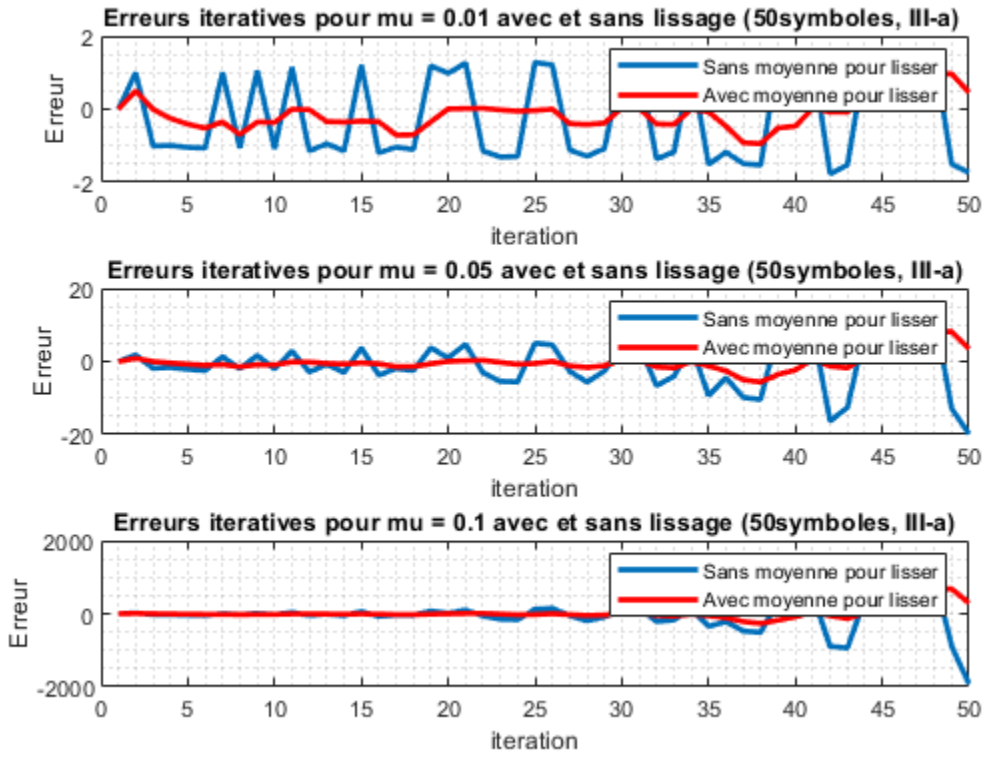
```
            w=w-mu*[y_list(i); y_list(i-1);
 y_list(i-2)]*(erreur_iterative);
        end
        erreur_iterative_list(i,compteur_mu) = erreur_iterative;

        % trouver s apres module de decision :
        if s_chapeau(i,compteur_mu) > 0
            s_guess(i,compteur_mu) = -1;
        else
            s_guess(i,compteur_mu) = 1;
        end

    end
    % faire la moyenne des derniers elements du vecteurs d'erreur
    % pour lisser la courbe :
    vecteur_a_moyenner = erreur_iterative_list(:,compteur_mu);
    % Nombre d'elements dans le vecteur
    n = length(vecteur_a_moyenner);
    % Nombre d'elements a considerer avant chaque element a moyenner
    elements_avant = 5;
    % Calcul des moyennes pour chaque ?l?ment avec les cinq pr?c?dents
    for k = 1:n
        % Indices des ?l?ments ? consid?rer
        indices = max(1, k - elements_avant):k;

        % Calcul de la moyenne
        moyennes_erreur_iterative_list(k, compteur_mu) =
 mean(vecteur_a_moyenner(indices));
    end
    compteur_mu = compteur_mu + 1;
end

% Calcul mmse question b
diff_carree = (s_guess(:,1) - s_envoye(:,1)).^2;
mmse_50symboles = mean(diff_carree);

% Calcul taux d'erreur par symbole sur les symboles envoy?s
difference_per_symbol = nnz(s_guess(:,1) ~= s_envoye(:,1)); % mu =
 0.01
taux_erreur_par_symbole_50symboles = difference_per_symbol/
dataset_size;



% PLOT L'EVOLUTION DE L'APPRENTISSAGE :
x = 1:1:dataset_size;
fig1 = figure();
subplot(3,1,1);
plot(x,erreur_iterative_list(:,1), 'linewidth', 2);
hold on
plot(x,moyennes_erreur_iterative_list(:,1),'-r', 'linewidth', 2);
grid minor
title('Erreurs iteratives pour mu = 0.01 avec et sans lissage
 (50symboles, III-a) ')
```

```
xlabel('iteration')
ylabel('Erreur');
legend('Sans moyenne pour lisser', 'Avec moyenne pour lisser')

subplot(3,1,2);
plot(x,erreur_iterative_list(:,2), 'linewidth', 2);
hold on
plot(x,moyennes_erreur_iterative_list(:,2),'-r', 'linewidth', 2);
grid minor
title('Erreurs iteratives pour mu = 0.05 avec et sans lissage
 (50symboles, III-a)')
xlabel('iteration')
ylabel('Erreur');
legend('Sans moyenne pour lisser', 'Avec moyenne pour lisser')


subplot(3,1,3);
plot(x,erreur_iterative_list(:,3), 'linewidth', 2);
hold on
plot(x,moyennes_erreur_iterative_list(:,3),'-r', 'linewidth', 2);
grid minor
title('Erreurs iteratives pour mu = 0.1 avec et sans lissage
 (50symboles, III-a)')
xlabel('iteration')
ylabel('Erreur');
legend('Sans moyenne pour lisser', 'Avec moyenne pour lisser')
```



# Export de la figure (automatique, eh oui!)

```
%1) en format vectoriel .eps pour les vrais qui utilisent LaTeX
print(fig1,'-depsc',strcat(path,nom_image,'.eps'))
%2) en format PNG (haute resolution 600dpi, pour ceux qui utilisent
 Word)
print(fig1,'-dpng','-r600',strcat(path,nom_image,'.png'))
```




# Egalisateur LMS (entrainement avec 500 symboles)

```
dataset_size = 500; %% INPUT, A MODIFIER
w=zeros(3,1);
y_list = y(:);
vecteurs_aleatoires_envoyes_list = vecteurs_aleatoires_envoyes(:);
erreur_iterative_list = 233423411*ones(dataset_size,3);
compteur_mu=1;
% Initialisation du vecteur pour stocker les moyennes
moyennes_erreur_iterative_list = zeros(dataset_size, 3);
s_chapeau = 34342342*ones(dataset_size,3);
s_guess = 34342342*ones(dataset_size,3);
s_envoye = vecteurs_aleatoires_envoyes_list(1:dataset_size);
for mu=[0.01 0.05 0.1]
    for i=1:1:dataset_size
```

```
        if i == 1
            s_chapeau(i,compteur_mu) = [y_list(i), 0, 0]*w;
            erreur_iterative = 0 - [y_list(i), 0, 0]*w;
            w=w-mu*[y_list(i); 0; 0]*(erreur_iterative);
        end
        if i == 2
            s_chapeau(i,compteur_mu) = [y_list(i), y_list(i-1), 0]*w;
            erreur_iterative = vecteurs_aleatoires_envoyes_list(i-1) -
 [y_list(i), y_list(i-1), 0]*w;
            w=w-mu*[y_list(i); y_list(i-1); 0]*(erreur_iterative);
        end
        if i > 2
            s_chapeau(i,compteur_mu) = [y_list(i), y_list(i-1),
 y_list(i-2)]*w;
            erreur_iterative = vecteurs_aleatoires_envoyes_list(i-1) -
 [y_list(i), y_list(i-1), y_list(i-2)]*w;
            w=w-mu*[y_list(i); y_list(i-1);
 y_list(i-2)]*(erreur_iterative);
        end
        erreur_iterative_list(i,compteur_mu) = erreur_iterative;

        % trouver s apres module de decision :
        if s_chapeau(i,compteur_mu) > 0
            s_guess(i,compteur_mu) = -1;
        else
            s_guess(i,compteur_mu) = 1;
        end

    end
    % faire la moyenne des derniers elements du vecteurs d'erreur
    % pour lisser la courbe :
    vecteur_a_moyenner = erreur_iterative_list(:,compteur_mu);
    % Nombre d'elements dans le vecteur
    n = length(vecteur_a_moyenner);
    % Nombre d'elements a considerer avant chaque element a moyenner
    elements_avant = 5;
    % Calcul des moyennes pour chaque ?l?ment avec les cinq pr?c?dents
    for k = 1:n
        % Indices des ?l?ments ? consid?rer
        indices = max(1, k - elements_avant):k;

        % Calcul de la moyenne
        moyennes_erreur_iterative_list(k, compteur_mu) =
 mean(vecteur_a_moyenner(indices));
    end
    compteur_mu = compteur_mu + 1;
end

% Calcul mmse question b
diff_carree = (s_guess(:,1) - s_envoye(:,1)).^2;
mmse_500symboles = mean(diff_carree);
disp("mmse_50symboles");
disp(mmse_50symboles);
disp("mmse_500symboles");
```

```
disp(mmse_500symboles);

% Calcul taux d'erreur par symbole sur les symboles envoy?s
difference_per_symbol = nnz(s_guess(:,1) ~= s_envoye(:,1)); % mu =
 0.01
taux_erreur_par_symbole_500symboles = difference_per_symbol/
dataset_size;
disp("taux_erreur_par_symbole_50symboles");
disp(taux_erreur_par_symbole_50symboles);
disp("taux_erreur_par_symbole_500symboles");
disp(taux_erreur_par_symbole_500symboles);

mmse_50symboles
     2

mmse_500symboles
    1.7440

taux_erreur_par_symbole_50symboles
    0.5000

taux_erreur_par_symbole_500symboles
    0.4360
```

# Commentaires question b)

Nous remarquons que nous avons des performances moins bonnes en entrainant avec 500 symboles plutot que 50 symboles (mmse et taux d'erreur plus eleve). L'entrainement semble moins performant que les m?thodes utilis?es dans les parties pr?c?dantes. Au vu de la difference entre les taux d'erreurs ici, en comparaison aux taux d'erreurs des parties I et II, il pourrait etre interessant d'essayer un module de decision autre (on pense notamment au module de decision DFS).

*Published with MATLAB® R2019b*