Abderrahmane Ouarach

La Théorie de l’Information et du Codage
Série de Travaux Pratiques
Filières : SUD INE1

- **PARTIE 1**

1. Génération d’un vecteur Z

```
% Taille du vecteur
n = 100;

% Génération du vecteur Z
Z = randi([0, 15], 1, n);

% Affichage du vecteur Z
disp(Z);
```

2. Calcul des probabilités des valeurs du vecteur Z

```
% Utilisation de histc pour calculer les fréquences
edges = 0:15; % Bornes des valeurs
counts = histc(Z, edges); % Compte le nombre d’occurrences de

% Calcul des probabilités
probabilities = counts / n;

% Affichage
```

```
disp('Probabilités des valeurs :');
disp(probabilities);
```

3. Calcul de l’entropie de Z

```
% Filtrer les probabilités non nulles
non_zero_probs = probabilities(probabilities > 0);

% Calcul de l'entropie
entropy_measured = -sum(non_zero_probs .* log2(non_zero_probs

% Affichage
disp(['Entropie mesurée de Z : ', num2str(entropy_measured)])
```

4. Entropie théorique d’une variable uniforme

```
% Entropie théorique
N = 16; % Nombre de valeurs possibles
entropy_theoretical = log2(N);

% Affichage
disp(['Entropie théorique : ', num2str(entropy_theoretical)])
disp(['Différence entre entropie mesurée et théorique : ', nu
```

5. Variation du nombre n d’échantillons et mesure de l’entropie

```
sample_sizes = [10, 50, 100, 500, 1000]; % Tailles différente
entropy_values = zeros(1, length(sample_sizes));

for idx = 1:length(sample_sizes)
    n = sample_sizes(idx);
    Z = randi([0, 15], 1, n);
    counts = histc(Z, edges);
    probabilities = counts / n;
    non_zero_probs = probabilities(probabilities > 0);
    entropy_values(idx) = -sum(non_zero_probs .* log2(non_zer
end
```

```
% Affichage des entropies
disp('Entropies mesurées pour différentes tailles d’échantill
disp(array2table([sample_sizes' entropy_values'], 'VariableNa
```

6. Simulation d’une variable binaire

```
N = 10^4; % Nombre d’échantillons
p = 0.3;  % Probabilité de succès

% Simulation de la variable binaire
X = rand(1, N) < p;

% Comptage des valeurs
h = hist(X, [0, 1]);

% Probabilités mesurées
P_0 = h(1) / N; % P(X=0)
P_1 = h(2) / N; % P(X=1)

% Affichage
disp(['P(X=0) : ', num2str(P_0)]);
disp(['P(X=1) : ', num2str(P_1)]);
```

7. Simulation d’une variable discrète avec une loi donnée

```
X = [0, 1, 2, 3, 4]; % Valeurs possibles
p = [0.3, 0.3, 0.2, 0.1, 0.1]; % Probabilités associées
F = cumsum(p); % Somme cumulative des probabilités

N = 10^2; % Taille de l’échantillon
r = rand(1, N); % Valeurs uniformes aléatoires entre 0 et 1
Xg = zeros(1, N);

% Génération des valeurs
for i = 1:N
    pos = find(r(i) < F, 1);
    Xg(i) = X(pos);
end
```

```
% Calcul des probabilités mesurées
ps = hist(Xg, X) / N;

% Affichage des résultats
figure;
stem(X, ps, 'x', 'DisplayName', 'Probabilités mesurées');
hold on;
stem(X, p, 'r', 'DisplayName', 'Probabilités théoriques');
legend;
xlabel('Valeurs de X');
ylabel('Probabilités');
title('Simulation d’une variable discrète');
```

- **PARTIE 2**

1. Utilisation de `huffmanenco` et `huffmandeco`

```
help huffmanenco
help huffmandeco
```

2. Exemple avec les symboles et probabilités

```
% Définition des symboles et des probabilités
symbols = 1:6;
p = [0.5, 0.125, 0.125, 0.125, 0.0625, 0.0625];

% Calcul des bits par symbole (théorique)
bps = ceil(log2(max(symbols)));
disp(['Bits par symbole : ', num2str(bps)]);
```

3. Création d’un dictionnaire Huffman

```
% Création du dictionnaire de Huffman
dict = huffmandict(symbols, p);

% Affichage du dictionnaire
```

```
disp('Dictionnaire Huffman :');
disp(dict);
```

4. Génération d’un vecteur de symboles aléatoires

```
% Génération d'un vecteur aléatoire de symboles
inputSig = randsrc(100, 1, [symbols; p]);

% Affichage de l'entrée
disp('Signal d’entrée :');
disp(inputSig');
```

5. Codage du message

```
% Encodage Huffman
code = huffmanenco(inputSig, dict);

% Affichage du code binaire
disp('Code binaire Huffman :');
disp(code');
```

6. Décodage et vérification

```
% Décodage Huffman
sig = huffmandeco(code, dict);

% Vérification
is_equal = isequal(inputSig, sig);

% Résultat
disp(['Le signal décodé est identique au signal initial : ',
```

7. Programme complet

```
% Définition des symboles et des probabilités
symbols = 1:6;
p = [0.5, 0.125, 0.125, 0.125, 0.0625, 0.0625];
```

```
% Création du dictionnaire de Huffman
dict = huffmandict(symbols, p);

% Génération d'un vecteur aléatoire de symboles
inputSig = randsrc(100, 1, [symbols; p]);

% Encodage Huffman
code = huffmanenco(inputSig, dict);

% Décodage Huffman
sig = huffmandeco(code, dict);

% Vérification
is_equal = isequal(inputSig, sig);

% Affichage des résultats
disp('Signal d’entrée :');
disp(inputSig');
disp('Code binaire Huffman :');
disp(code');
disp('Signal décodé :');
disp(sig');
disp(['Le signal décodé est identique au signal initial : ',
```

1. **Signal d’entrée** : Une séquence aléatoire de 100 valeurs parmi {1, 2, 3, 4, 5, 6}.
2. **Code Huffman** : Une séquence binaire plus compacte que l’entrée.
3. **Vérification** : Le signal décodé sera identique au signal original.

- **PARTIE 3**

1. Générer le message binaire `msg`

```
Nb = 100; % Nombre de bits
msg = randint(Nb, 1, [0 1]); % Message binaire aléatoire (0 o
disp('Message généré :');
disp(msg');
```

2. Codage par répétition (3,1)

```
M = 3; % Nombre de répétitions
code_vector = repmat(msg, 1, M); % Codage par répétition
disp('Message codé :');
disp(code_vector);
```

3. Simuler un canal binaire symétrique (BSC)

```
p = 10^(-3); % Probabilité d’erreur du canal
bsc_code_vector = zeros(size(code_vector)); % Initialisation

for n = 1:Nb
    for m = 1:M
        z = rand(1, 1); % Générer un nombre aléatoire entre 0
        if z < p % Introduire une erreur avec probabilité p
            bsc_code_vector(n, m) = not(code_vector(n, m));
        else
            bsc_code_vector(n, m) = code_vector(n, m);
        end
    end
end

disp('Vecteur à la sortie du canal :');
disp(bsc_code_vector);
```

4. Décoder la séquence reçue

```
% Conversion en matrice
decoded_matrix = vec2mat(bsc_code_vector(:), M);

% Décodage par décision majoritaire
decoded_msg = zeros(1, Nb);
for n = 1:Nb
    if sum(decoded_matrix(n, :)) < M/2 % Majorité de 0
        decoded_msg(n) = 0;
    else % Majorité de 1
        decoded_msg(n) = 1;
```

```
    end
end

disp('Message décodé :');
disp(decoded_msg);
```

5. Calcul de la probabilité d’erreur avec et sans codage

```
Nb_sim = 10000; % Nombre de blocs simulés
errors_with_coding = 0;
errors_without_coding = 0;

for k = 1:Nb_sim
    % Générer un nouveau message
    msg = randint(Nb, 1, [0 1]);
    code_vector = repmat(msg, 1, M);

    % Canal BSC
    bsc_code_vector = zeros(size(code_vector));
    for n = 1:Nb
        for m = 1:M
            z = rand(1, 1);
            if z < p
                bsc_code_vector(n, m) = not(code_vector(n, m)
            else
                bsc_code_vector(n, m) = code_vector(n, m);
            end
        end
    end

    % Décodage
    decoded_matrix = vec2mat(bsc_code_vector(:), M);
    decoded_msg = zeros(1, Nb);
    for n = 1:Nb
        if sum(decoded_matrix(n, :)) < M/2
            decoded_msg(n) = 0;
        else
            decoded_msg(n) = 1;
```

```
        end
    end

    % Calcul des erreurs
    errors_with_coding = errors_with_coding + sum(decoded_msg
    errors_without_coding = errors_without_coding + sum(randi
end

% Probabilité d'erreur
P_error_with_coding = errors_with_coding / (Nb_sim * Nb);
P_error_without_coding = errors_without_coding / (Nb_sim * Nb

disp(['Probabilité d'erreur avec codage : ', num2str(P_error_
disp(['Probabilité d'erreur sans codage : ', num2str(P_error_
```

6. Simulation d'une variable binaire

```
N = 10^4; % Nombre d'échantillons
p = 0.3;  % Probabilité de succès

% Simulation de la variable binaire
X = rand(1, N) < p;

% Comptage des valeurs
h = hist(X, [0, 1]);

% Probabilités mesurées
P_0 = h(1) / N; % P(X=0)
P_1 = h(2) / N; % P(X=1)

% Affichage
disp(['P(X=0) : ', num2str(P_0)]);
disp(['P(X=1) : ', num2str(P_1)]);
```

7. Simulation d'une variable discrète avec une loi donnée

```
X = [0, 1, 2, 3, 4]; % Valeurs possibles
p = [0.3, 0.3, 0.2, 0.1, 0.1]; % Probabilités associées
```

```
F = cumsum(p); % Somme cumulative des probabilités

N = 10^2; % Taille de l’échantillon
r = rand(1, N); % Valeurs uniformes aléatoires entre 0 et 1
Xg = zeros(1, N);

% Génération des valeurs
for i = 1:N
    pos = find(r(i) < F, 1);
    Xg(i) = X(pos);
end

% Calcul des probabilités mesurées
ps = hist(Xg, X) / N;

% Affichage des résultats
figure;
stem(X, ps, 'x', 'DisplayName', 'Probabilités mesurées');
hold on;
stem(X, p, 'r', 'DisplayName', 'Probabilités théoriques');
legend;
xlabel('Valeurs de X');
ylabel('Probabilités');
title('Simulation d’une variable discrète');
```

Résumé des résultats

1. **Codage à répétition (3,1)** réduit la probabilité d’erreur par la redondance et le vote majoritaire.
2. **Avec codage** : La probabilité d’erreur devrait être beaucoup plus faible.
3. **Sans codage** : L’erreur est uniquement liée à la probabilité p.