

Le Mans Université

Licence Informatique *3ème année*
Module de Codage de l'Information

COMPTE RENDU DE TP

PENNACHI Cyprien, TAN Tony

7 novembre 2022

Table des matières

1	Introduction	3
2	HDBn	4
2.1	Encodage	4
2.2	Décodage	4
3	Arithmétique	5
3.1	Encodage	5
3.2	Décodage	5
3.3	Problèmes	6
4	Séquence d'étalonnage	7
4.1	Création de la matrice de Hadamard	7
4.2	Encodage	7
4.3	Décodage	7
5	Codage Longueur Maximale	8
5.1	Codage Gold	8
5.2	Codage JPL	8
6	Annexe	10

1 Introduction

Cette introduction présentera le cadre et le but du document, le sujet qui sera traité et une présentation du plan adopté.

Ce document a pour objet de présenter le travail réalisé dans le cadre du module de Codage de l'information de la formation de licence 3 Informatique de l'Université du Mans. Il a pour objet de présenter les différents moyens de coder l'information vu en cours.

Dans ce document nous présenterons dans une première partie le code d'étalement avec les encodeurs/décodeurs HDBn et Arithmétique puis la matrice d'Hadamard, ensuite dans une seconde partie nous aborderons les différents générateurs pseudo aléatoire vu en cours avec la génération d'un codeur à longueur maximal, la mise en oeuvre d'un codeur Gold et JPL.

Pour tester notre programme nous avons fait une arborescence de fichier avec les fichiers .c dans le dossier src/ les .h dans lib/ les .o dans obj/ et les programmes dans bin/ avec un joli makefile. Il suffira d'appuyer sur "make clean" puis "make all" pour compiler tous les fichiers test. Vous pourrez ensuite lancer les programmes :

- ./hadamard.exe Pour le code d'étalement
- ./hdbn_arith Pour le codage HDBn et le codage arithmétique
- ./gen_pseudo_alea.exe Pour le générateur pseudo aléatoire

2 HDBn

2.1 Encodage

Pour le codage HDBn, l'utilisateur choisit le n.

La fonction de codage prend en paramètre la valeur du n, un message pour renvoyer le message encodé.

Ses variables sont une polarité qui est un entier, cela servira à changer la polarité en multipliant par -1 à la puissance de la variable polarité, un compteur pour chaque zéro consécutif, une variable de type booléenne qui vérifie la polarité du dernier viol.

L'utilisateur choisit un message qu'on supposera binaire(0 ou 1). Le message est ensuite traité.

Pour le traitement, on regarde chaque chiffre pour connaître son état.

Si le chiffre est un 1, alors on multiplie ce 1 par -1 avec la polarité en puissance. Le compteur de zéro est remis à 0.

Si, au contraire, le chiffre est à 0, on incrémente de 1 la valeur du compteur de zéro. - Si le compteur de zéro est égal à la valeur du n, il faut regarder la polarité du viol. -La première étape est de remettre le compteur de zéro à 0. Si la polarité du viol est positive et que le dernier 1 est positif, alors la valeur du chiffre en n position avant la position du chiffre actuelle passe à -1. La valeur du chiffre actuelle passe aussi à -1. Sinon la valeur du dernier 1 est négatif, alors seulement le chiffre actuel passe à -1.

Sinon, si la polarité du viol est négative et que le dernier 1 est négatif, alors la valeur du chiffre en n position avant la position du chiffre actuelle passe à 1. La valeur du chiffre actuelle passe aussi à 1. Sinon la valeur du dernier 1 est positif, alors seulement le chiffre actuel passe à 1. schéma de l'algorithme dans Annexe page 10, Figure 2.

2.2 Décodage

Pour le décodage, la fonction passe en paramètre le message encodé, le n ainsi que la taille du message.

Pour chaque chiffre, on vérifie :

- Le nombre de zéro
- La valeur du chiffre
- La polarité du dernier viol

Si un chiffre est égal à 1 et que le compteur de zéro n'est pas égale à n-1 alors ça ne change pas. Si, au contraire, le compteur est égal à n-1 et que le dernier 1 était positif alors le chiffre actuel passe à 0. Si, au contraire, le compteur est égal à n-1 et que le dernier -1 était négatif alors le chiffre actuel passe à 0 et le n chiffre avant passe à 0.

Si un chiffre est égal à -1 et que le compteur de zéro n'est pas égale à n-1 alors le chiffre passe à 1. Si, le compteur est égal à n-1 et que le dernier 1 était négatif alors le chiffre actuel passe à 0. Si, au contraire, le compteur est égal à

n-1 et que le dernier 1 était positif alors le chiffre actuel passe à 0 et le n chiffre avant passe à 0.

3 Arithmétique

Le principe du codage arithmétique est de compresser un message de façon à ce que le récepteur ait un nombre décimal compris entre 0 et 1 à décoder.

3.1 Encodage

Pour compresser le message, j'utilise la fonction `codage_arithmetique` qui prend en paramètre un tableau pour avoir l'intervalle du mot compressé. Cette fonction prend aussi un tableau de type hash (une structure que j'ai créé qui comporte un symbole, un compteur et les intervalles du symbole voir Annexe page 11, Figure 3).

Je commence par demander le message de l'utilisateur.

Ensuite il faut compter le nombre d'occurrences de chaque élément et calculer leur intervalle. Donc je regarde pour chaque élément si il est nouveau ou non. Si il est nouveau, je le place dans le buffer `tab_symbole` (qui sera le message sans occurrence). Sinon, j'incrémente seulement le compteur de l'élément.

On calcule ensuite la probabilité de chaque élément en divisant le nombre d'occurrence de l'élément par le nombre total d'élément. Je crée ensuite les intervalles pour chaque élément. Pour la borne inférieure de chaque élément, je prends la borne supérieure de l'élément précédent de `tab_symbole`. Concernant la borne supérieure, on ajoute la borne inférieure de l'élément à la probabilité.

Une fois les intervalles réalisés de chaque élément, j'effectue ces calculs pour calculer les intervalles du mot compressé et ainsi réduire l'intervalle du mot.

$BS = BI + (BS - BI) * (BS_symbole)$

$BI = BI + (BS - BI) * (BI_symbole)$

Avec BS = borne supérieure et BI = borne inférieure. Ainsi le mot compressé sera dans l'intervalle de ces deux bornes.

3.2 Décodage

Pour décompresser le mot, je choisis arbitrairement un nombre qui sera égale à la borne inférieure du mot compressé (pour des soucis de taille du nombre que le langage C ne peut régler) pour re-obtenir notre mot de départ et pour cela j'utilise ma fonction `decodage_arithmetique` qui prend en paramètre les bornes calculés dans le codeur, le nombre choisi, la taille du message, le tableau de symbole sans occurrence et le nombre de symboles du tableau de symbole sans occurrence. Je vérifie si le nombre est dans l'intervalle d'un des symboles du tableau de symbole si il n'y est pas ça veut dire que y'a un petit soucis dans le codeur mais si il y est on effectue le calcul suivant : $n = (n - BI_symbole) / p$ avec n = le nombre et p = la probabilité du symbole. Puis on refait jusqu'à le

faire taille du message fois. On se retrouve ensuite avec notre message initiale.
exemple Annexe page 12, Figure 4

3.3 Problèmes

Le problème qui peut y avoir c'est lorsque le message devient de plus en plus grand, les bornes du messages deviendront de plus en plus petit jusqu'à qu'il n'y ait pas moyen de l'afficher pour le compilateur C et du coup des erreurs de décodage peut se produire au niveau des intervalles. exemple Annexe page 13, Figure 5

4 Séquence d'étalonnement

4.1 Création de la matrice de Hadamard

Pour la création de la matrice de Hadamard, je demande à l'utilisateur le nombre d'utilisateurs souhaitant utiliser la séquence d'étalonnement puis avec ce nombre je construis la matrice avec ma fonction *generation_matrice*.

Ce qui a été le plus dur c'était de calculer la taille de la matrice puisque ma matrice est statique et non dynamique (cela évite des fuites et des pertes de mémoire avec *realloc*) donc avec le nombre d'utilisateurs j'effectue *taille_matrice* pour avoir la taille de la matrice et *longueur_matrice* pour avoir la longueur de la matrice. Avec cette longueur je peux créer ma variable **matrice[longueur_matrice()][longueur_matrice()]** qui contiendra la matrice de Hadamard.

La fonction pour générer la matrice *generation_matrice* est récursive elle prend en paramètre la matrice pour la modifier et la taille de la matrice que je diminuerai jusqu'à obtenir la taille 0. Mon algorithme est la suivante la fonction s'appelle elle même en diminuant à chaque fois la taille de la matrice jusqu'à atteindre 0 puis je remplis ma matrice jusqu'à revenir à ma taille initiale et normalement elle est totalement remplie.

4.2 Encodage

Après avoir créer ma matrice, je demande à l'utilisateur la taille que les messages devront respecter. Ensuite je demande pour chaque utilisateur un message (ça doit être une suite binaire bien sûr) et une ligne de la matrice de Hadamard. En multipliant chaque bit de la ligne de la matrice et chaque bit du message, une séquence est générée avec la fonction *generation_sequence*, donc je fais cela avec tous les messages. Avec toutes ces séquences j'effectue l'étalonnement c'est à dire j'additionne mes séquences (elle est effectuée à la fin de ma fonction *initialisation_sequence*). un exemple d'utilisation avec 3 utilisateurs est réalisé à la page 14, dans la figure n°6.

4.3 Décodage

Pour le décodage, je n'ai pas réussi à le coder car je ne sais pas quel calcul faire pour retrouver le message avec le résultat de l'étalonnement. Même en stockant les messages ainsi que les lignes de la matrice de Hadamard je ne vois pas.

5 Codage Longueur Maximale

Avant d'avoir un code à Longueur Maximal(L.M), je demande à l'utilisateur un nombre d'étage n avec ma fonction *saisie_nb_etages*, un vecteur avec ma fonction *saisie_vecteur* qui vérifie si le vecteur est bien une suite binaire, et un polynome avec ma fonction *saisie_polynome*.

Pour avoir le polynome, un fichier **polynome.txt** a été créé dans le dossier lib contenant tous les polynomes de chaque nombres d'étages (une ligne = un nombre d'étage) jusqu'à $n=34$ et ma fonction *saisie_polynome* appelle la fonction *remplir_matrice_polynome* qui lui permet de stocker tous les polynomes d'une ligne dans mon tableau de polynomes soit une matrice de polynome. l'utilisateur n'aura plus qu'à donner le n° du polynome souhaiter pour obtenir son polynome.

Avec le nombre d'étages, le vecteur et le polynome, je peux effectuer le codeur à L.M avec ma fonction *codeur_longueur_maximal* qui les prend en paramètres avec un tableau pour le résultat du codeur. Dans cette fonction je calcule la longueur séquentielle L (j'aurai dû le donner en paramètre mais y'avait beaucoup trop de paramètre donc à la place je renvoie la longueur L en sortie de fonction) puis je fais une combinaison de L registres avec le vecteur et dans cette combinaison je calcule le bit XOR en parcourant le polynôme et avec cela je décale les bits du vecteur vers la droite et je mets le bit XOR en première position du vecteur en stockant dans le résultat le bit qui disparaît. Ensuite j'affiche le résultat. exemple de code à la page 15, dans la figure 7.

5.1 Codage Gold

Avant de faire le codage Gold, je demande à l'utilisateur de faire un autre code à L.M avec le même nombre d'étages n . Puis je peux faire le codeur Gold avec la fonction *codeur_gold*. Dans cette fonction je lui passe en paramètre le 1er code à L.M et le 2ème code à L.M et j'effectue un XOR pour chaque bit des deux codes et j'affiche le résultat. exemple de code à la page 16, dans la figure 8.

5.2 Codage JPL

Avant de faire le codage JPL, je demande à l'utilisateur d'effectuer deux codes L.M supplémentaires pour effectuer le codage JPL avec la fonction *codeur_JPL* mais d'abord je vérifie si les longueurs séquentielles sont premiers entre eux.

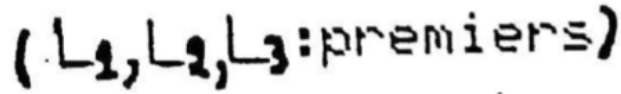
A handwritten note in black ink on a white background. The text is enclosed in parentheses and reads "(L1, L2, L3: premiers)". The handwriting is somewhat cursive and slightly blurred.

FIGURE 1 – L1,L2 et L3 doit être premiers entre eux pour utiliser le code JPL

Si ils ne le sont pas, je redemande à l'utilisateur de re-effectuer les deux codes L.M supplémentaires jusqu'à que L1,L2 et L3 soient premiers entre eux. Pour savoir si ils sont premiers, j'ai créé une fonction *pgcd* qui calcule le plus grand diviseur commun entre ces 3 longueurs et si la fonction renvoie 1 cela veut dire qu'ils sont premiers entre eux. C'est après tout cette procédure que la fonction *codeur_JPL* est utilisée pour le codage JPL. Cette fonction va juste prendre en paramètre les résultats des 3 codes à L.M et d'abord faire un XOR entre les deux premiers puis un XOR avec le résultat du premier XOR et le dernier code et enfin je l'affiche. On peut en trouver un exemple à la page 17,dans la figure 9.

6 Annexe

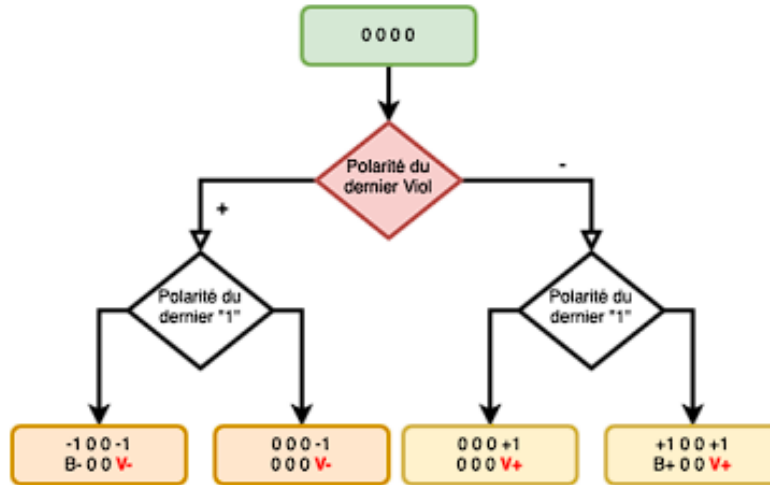


FIGURE 2 – Schéma de choix concernant le bit de viol selon le dernier 1

```
typedef struct{  
    char symbole;  
    int cpt;  
    double BI;  
    double BS;  
}hash;
```

FIGURE 3 – Structure nommé hash composé d'un symbole,un compteur et des intervalles

```

$ ./bin/hdbn.arith.exe
1 pour codage arithmétique.
2 pour le codage HDBn, vous devez choisir 1 ou 2 :
1
vous avez choisi le codage Arithmétique.
Saisir un message pour arithmétique (Appuyez sur entrée pour finir votre message!):
automne
votre message : automne
nb_occu pour a: 1 intervalle=[0.000000,0.142857]
nb_occu pour u: 1 intervalle=[0.142857,0.285714]
nb_occu pour t: 1 intervalle=[0.285714,0.428571]
nb_occu pour o: 1 intervalle=[0.428571,0.571429]
nb_occu pour m: 1 intervalle=[0.571429,0.714286]
nb_occu pour n: 1 intervalle=[0.714286,0.857143]
nb_occu pour e: 1 intervalle=[0.857143,1.000000]

borne inf : 0.000000 borne sup : 0.142857
borne inf : 0.020408 borne sup : 0.040816
borne inf : 0.026239 borne sup : 0.029155
borne inf : 0.027489 borne sup : 0.027905
borne inf : 0.027727 borne sup : 0.027786
borne inf : 0.027769 borne sup : 0.027778
borne inf : 0.027776 borne sup : 0.027778
le mot compressé est donc compris entre les bornes 0.027776 et 0.027778

Passage au decodage

Le message compressé étant compris dans un intervalle très petit il sera automatiquement égal à la borne inférieure soit 0.027776
taille=7
Valeur du message compressé compris dans l'intervalle [0.000000,0.142857] de la lettre a
Après calcul, nouvelle valeur du message_compressé : 0.194434
Valeur du message compressé compris dans l'intervalle [0.142857,0.285714] de la lettre u
Après calcul, nouvelle valeur du message_compressé : 0.361040
Valeur du message compressé compris dans l'intervalle [0.285714,0.428571] de la lettre t
Après calcul, nouvelle valeur du message_compressé : 0.527280
Valeur du message compressé compris dans l'intervalle [0.428571,0.571429] de la lettre o
Après calcul, nouvelle valeur du message_compressé : 0.690962
Valeur du message compressé compris dans l'intervalle [0.571429,0.714286] de la lettre m
Après calcul, nouvelle valeur du message_compressé : 0.836735
Valeur du message compressé compris dans l'intervalle [0.714286,0.857143] de la lettre n
Après calcul, nouvelle valeur du message_compressé : 0.857143
Valeur du message compressé compris dans l'intervalle [0.857143,1.000000] de la lettre e
Après calcul, nouvelle valeur du message_compressé : 0.000000
Mot decodé : automne
FIN DU DECODAGE

```

FIGURE 4 – Codage arithmétique avec le mot "automne"

```

$ ./bin/hdbn_arith.exe
1 pour codage arithmétique.
2 pour le codage Hdbn.vous devez choisir 1 ou 2 :
1
vous avez choisi le codage Arithmétique.
Saisir un message pour arithmetique (Appuyez sur entr[ée pour finir votre message]):
electricite
Votre message : electricite
nb_occu pour e: 3 intervalle=[0.000000,0.272727]
nb_occu pour l: 1 intervalle=[0.272727,0.363636]
nb_occu pour c: 2 intervalle=[0.363636,0.545455]
nb_occu pour t: 2 intervalle=[0.545455,0.727273]
nb_occu pour r: 1 intervalle=[0.727273,0.818182]
nb_occu pour i: 2 intervalle=[0.818182,1.000000]

borne inf : 0.000000 borne sup : 0.272727
borne inf : 0.074380 borne sup : 0.099174
borne inf : 0.074380 borne sup : 0.081142
borne inf : 0.076839 borne sup : 0.078068
borne inf : 0.077510 borne sup : 0.077733
borne inf : 0.077672 borne sup : 0.077692
borne inf : 0.077689 borne sup : 0.077692
borne inf : 0.077690 borne sup : 0.077691
borne inf : 0.077691 borne sup : 0.077691
borne inf : 0.077691 borne sup : 0.077691
borne inf : 0.077691 borne sup : 0.077691
Le mot compressé est donc compris entre les bornes 0.077691 et 0.077691

Passage au decodage
Le message compressé étant compris dans un intervalle très petit il sera automatiquement égale à la borne inférieure soit 0.077691
taille=11
Valeur du message compressé compris dans l'intervalle [0.000000,0.272727] de la lettre e
Après calcul, nouvelle valeur du message_compressé : 0.284866
Valeur du message compressé compris dans l'intervalle [0.272727,0.363636] de la lettre l
Après calcul, nouvelle valeur du message_compressé : 0.133527
Valeur du message compressé compris dans l'intervalle [0.000000,0.272727] de la lettre e
Après calcul, nouvelle valeur du message_compressé : 0.489601
Valeur du message compressé compris dans l'intervalle [0.363636,0.545455] de la lettre c
Après calcul, nouvelle valeur du message_compressé : 0.692804
Valeur du message compressé compris dans l'intervalle [0.545455,0.727273] de la lettre t
Après calcul, nouvelle valeur du message_compressé : 0.810420
Valeur du message compressé compris dans l'intervalle [0.727273,0.818182] de la lettre r
Après calcul, nouvelle valeur du message_compressé : 0.914623
Valeur du message compressé compris dans l'intervalle [0.818182,1.000000] de la lettre i
Après calcul, nouvelle valeur du message_compressé : 0.530428
Valeur du message compressé compris dans l'intervalle [0.363636,0.545455] de la lettre c
Après calcul, nouvelle valeur du message_compressé : 0.917355
Valeur du message compressé compris dans l'intervalle [0.818182,1.000000] de la lettre i
Après calcul, nouvelle valeur du message_compressé : 0.545455
Valeur du message compressé compris dans l'intervalle [0.363636,0.545455] de la lettre c
Après calcul, nouvelle valeur du message_compressé : 1.000000
Valeur du message compressé compris dans l'intervalle [0.818182,1.000000] de la lettre i
Après calcul, nouvelle valeur du message_compressé : 1.000000
Mot decodé : electricite
FIN DU DECODAGE

```

FIGURE 5 – Codage arithmétique avec le mot "electricite"

```

$ ./bin/hadamard.exe
Bonjour, Veuillez saisir le nombre d'utilisateurs : 3
taille matrice =2
Generation de la matrice Hadamard en fonction du nb d'utilisateur de longueur 4

[ 1, 1, 1, 1]
[ 1,-1, 1,-1]
[ 1, 1,-1,-1]
[ 1,-1,-1, 1]
Saisissez une taille de message : 4
Utilisateur 1 Quelle sequence de la matrice voulez-vous ?(entre 1 et 4): 1

Veuillez ecrire votre message binaire de taille 4 :
1011

Utilisateur 2 Quelle sequence de la matrice voulez-vous ?(entre 1 et 4): 2

Veuillez ecrire votre message binaire de taille 4 :
0000

Utilisateur 3 Quelle sequence de la matrice voulez-vous ?(entre 1 et 4): 4

Veuillez ecrire votre message binaire de taille 4 :
1010

Codage du message code avec les sequences choisies :
utilisateur 1 : 1 1 1 1-1-1-1-1 1 1 1 1 1 1 1
utilisateur 2 :-1 1-1 1-1 1-1 1-1 1-1 1-1 1-1 1
utilisateur 3 : 1-1-1 1-1 1 1-1 1-1-1 1-1 1 1-1
Code apres etalonnment :
1,1,-1,3,-3,1,-1,-1,1,1,-1,3,-1,3,1,1

```

FIGURE 6 – Cas d'utilisation du programme hadamard avec 3 utilisateurs

```

$ ./bin/gen_pseudo_alea.exe

Combien voulez-vous d'etages ? : 3

Le nombre d'etages est egal a 3

Quel vecteur voulez-vous(suite de nombre binaire de longueur 3) : 101

Le vecteur est : 101
Voici tous les polynomes disponibles pour n=3 :
polynome 1 : [1,3]
Quelle polynome choisissez-vous (choisissez le numero)? : 1
Vous avez choisi le polynome n1
Soit la longueur sequentielle L1 est egale à 7
Nombre d'elements dans le polynome = 2
      DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 1 0 1
bit_XOR = 0
registre 2 -> 0 1 0
bit_XOR = 0
registre 3 -> 0 0 1
bit_XOR = 1
registre 4 -> 1 0 0
bit_XOR = 1
registre 5 -> 1 1 0
bit_XOR = 1
registre 6 -> 1 1 1
bit_XOR = 0
registre 7 -> 0 1 1
CODE FINAL
1101001

      FIN DU CODEUR A LONGUEUR MAXIMALE

Quel type de codeur voulez-vous ?
1 : Codeur Gold
2 : Codeur JPL
3 : Quittez
Pour choisir entre les trois,Taper 1, 2 ou 3 : 3
A bientot !

```

FIGURE 7 – Cas d'utilisation du programme pour faire un code à Longueur Maximal avec n=3

```

$ ./bin/gen_pseudo_alea.exe

Combien voulez-vous d'etages ? : 2

Le nombre d'etages est egal a 2

Quel vecteur voulez-vous(suite de nombre binaire de longueur 2) : 01

Le vecteur est : 01
Voici tous les polynomes disponibles pour n=2 :
polynome 1 : [1,2]
Quelle polynome choisissez-vous (choisissez le numero)? : 1
Vous avez choisi le polynome n1
Soit la longueur sequentielle L1 est egale à 3
Nombre d'elements dans le polynome = 2
      DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 0 1
bit_XOR = 1
registre 2 -> 1 0
bit_XOR = 1
registre 3 -> 1 1
CODE FINAL
110
      FIN DU CODEUR A LONGUEUR MAXIMALE

Quel type de codeur voulez-vous ?
1 : Codeur Gold
2 : Codeur JPL
3 : Quittez
Pour choisir entre les trois,Taper 1, 2 ou 3 : 1
Pour utiliser le codeur Gold il faudra effectuer un autre code a longueur maximal avec le meme nombre d'etage n.

Quel vecteur voulez-vous(suite de nombre binaire de longueur 2) : 11

Le vecteur est : 11
Voici tous les polynomes disponibles pour n=2 :
polynome 1 : [1,2]
Quelle polynome choisissez-vous (choisissez le numero)? : 1
Vous avez choisi le polynome n1
Nombre d'elements dans le polynome = 2
      DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 1 1
bit_XOR = 0
registre 2 -> 0 1
bit_XOR = 1
registre 3 -> 1 0
CODE FINAL
111
      FIN DU CODEUR A LONGUEUR MAXIMALE
      DEMARRAGE DU CODEUR GOLD
1er code a longueur maximale :
1 1 0
2eme code a longueur maximale :
1 1 1
code final :
0 0 1
      FIN DU CODEUR GOLD

```

FIGURE 8 – Cas d'utilisation du programme pour faire un code Gold avec deux code à Longueur Maximal n=3


```

$ ./bin/gen_pseudo_alea.exe
Combien voulez-vous d'etages ? : 2
Le nombre d'etages est egal a 2
Quel vecteur voulez-vous(suite de nombre binaire de longueur 2) : 11
Le vecteur est : 11
voici tous les polynomes disponibles pour n=2 :
polynome 1 : [1,2]
Quelle polynome choisissez-vous (choisissez le numero)? : 1
Vous avez choisi le polynome n1
Soit la longueur sequentielle L1 est egale à 3
Nombre d'elements dans le polynome = 2
DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 1 1
bit_XOR = 0
registre 2 -> 0 1
bit_XOR = 1
registre 3 -> 1 0
CODE FINAL
1111
FIN DU CODEUR A LONGUEUR MAXIMALE
Quel type de codeur voulez-vous ?
1 : Codeur Gold
2 : Codeur JPL
3 : Quittez
Pour choisir entre les trois,Taper 1, 2 ou 3 : 2
Pour utiliser le codeur JPL il faudra effectuer deux autres codes a longueur maximal et que n,m et p soit premier entre eux
Procedure pour le deuxieme code L2 a longueur maximal :
Combien voulez-vous d'etages ? : 3
Le nombre d'etages est egal a 3
Soit la longueur sequentielle est egale à 7
Quel vecteur voulez-vous(suite de nombre binaire de longueur 3) : 101
Le vecteur est : 101
voici tous les polynomes disponibles pour n=3 :
polynome 1 : [1,3]
Quelle polynome choisissez-vous (choisissez le numero)? : 1
Vous avez choisi le polynome n1
Nombre d'elements dans le polynome = 2

```

FIGURE 9 – Exemple du codeur JPL avec les nombres d'étages $n=2$ $m=3$ et $p=5$

```

      DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 1 0 1
bit_XOR = 0
registre 2 -> 0 1 0
bit_XOR = 0
registre 3 -> 0 0 1
bit_XOR = 1
registre 4 -> 1 0 0
bit_XOR = 1
registre 5 -> 1 1 0
bit_XOR = 1
registre 6 -> 1 1 1
bit_XOR = 0
registre 7 -> 0 1 1
CODE FINAL
1101001

      FIN DU CODEUR A LONGUEUR MAXIMALE
Procédure pour le troisième et dernier code à longueur maximale :

Combien voulez-vous d'étages ? : 5

Le nombre d'étages est égal à 5
Soit la longueur séquentielle L3 est égale à 31

Quel vecteur voulez-vous (suite de nombre binaire de longueur 5) : 11011

Le vecteur est : 11011
Voici tous les polynômes disponibles pour n=5 :
polynome 1 : [2,5]
polynome 2 : [2,3,4,5]
polynome 3 : [1,2,4,5]
Quelle polynome choisissez-vous (choisissez le numéro)? : 3
Vous avez choisi le polynome n3
Nombre d'éléments dans le polynome = 4
      DEMARRAGE DU CODEUR A LONGUEUR MAXIMALE
COMBINAISON DES REGISTRES
registre 1 -> 1 1 0 1 1
bit_XOR = 0
registre 2 -> 0 1 1 0 1
bit_XOR = 0
registre 3 -> 0 0 1 1 0
bit_XOR = 1
registre 4 -> 1 0 0 1 1
bit_XOR = 1
registre 5 -> 1 1 0 0 1
bit_XOR = 1
registre 6 -> 1 1 1 0 0
bit_XOR = 0
registre 7 -> 0 1 1 1 0
bit_XOR = 0
registre 8 -> 0 0 1 1 1
bit_XOR = 0
registre 9 -> 0 0 0 1 1
bit_XOR = 0
registre 10 -> 0 0 0 0 1

```

