Codage de l'information
M1 Informatique 2015/16
TP Code d'étalement

Letay Benoit

15 décembre 2015

1 Introduction

L'objectif de ce TP est d'écrire l'ensemble des fonctions composants la mise en oeuvre du codage de Hadamard.

2 Fonctionnalités

Notre programme génère une N utilisateurs qui envoient en même temps un message de longueur L à travers un canal parfait. Ces messages sont d'abord codé à l'aide à l'aide de séquences de Hadamard (générés par une matrice pouvant apporter une nombre de séquences non répétée supérieur ou égal au nombre d'utilisateurs) servant de clefs. Ces clefs servent ensuite à decoder le message une fois celui ci transmit à travers le canal.

3 Usage

Usage, nombre d'utilisateurs puis la taille des messages
Exemple d'utilisation : figure 1 page 4

4 Fonctionnement

La fonction matrixHadamar génère une matrice de Hadamard de manière récursive de taille n entré en parametre et la retourne.

```
1 def matrixHadamard(n):
2     if n == 0:
3         return  np.matrix([1])
4     matrix =  matrixHadamard(n-1)
5     return np.concatenate((np.concatenate((matrix,matrix),axis=1),
6     np.concatenate((matrix,np.negative(matrix)),axis = 1)))
```

La fonction matrixHadamard2 génère une matrice de Hadamard de manière récursive mais terminale cette fois ci, cependant en python cette méthode n'est pas conseillée, nous ne l'utilisons donc pas dans notre programme.

```
def matrixHadamardTerminal(n,matrix):
    if n == 0:
        return  matrix
    matrix = np.concatenate((np.concatenate((matrix,matrix),axis=1),
    np.concatenate((matrix,np.negative(matrix)),axis = 1)))
    return matrixHadamardTerminal(n-1,matrix)

def matrixHadamard2(n):
    return matrixHadamardTerminal(n,np.matrix([1]))
```

La fonction codage code un message transmit en paramètre à l'aide d'une clé elle aussi transmise en paramètre. Le codage consiste simplement à multiplier le message par la clé.

```
def codage(seq_id,message):
    coded_message = []
    for x in range(len(message)):
        for y in range(len(seq_id)):
            coded_message.append( message[x] * seq_id[y] )
    return coded_message
```

La fonction décodage décode un message transmit en paramètre à l'aide d'une clé elle aussi transmise en paramètre. Le décodage consiste à multiplier le message recu par sa clé puis à diviser chaque valeur par la longueur de cette clé.

```
def decodage(seq_id,message):
    decoded_message = []
    y = 0
    somme = 0.0
    for x in range(len(message)):
        somme += message[x] * seq_id[y]
        y += 1
        if y % (len(seq_id) ) == 0:
            decoded_message.append( int(somme/len(seq_id)))
            y = 0
            somme = 0.0
    return decoded_message
```

La fonction generatemessage permet de générer un message de taille L entré en paramètre de manière aléatoire (dans les limites de l'ordinateur), elle nous permet de tester nos fonctions de codage/décodage.

```
def generate_message(taille):
    message = []
    for x in range(0,taille):
        val = random.randint(0,1)
        if val == 0:
```

```
6                 message.append(-1)
7             else:
8                 message.append(1)
9         return message
```

5 Analyse des résultats

Le codage et le décodage se passe sans erreurs se qui est plutôt rassurant car nous n'utilisons pas de canal gaussien.Malgré un grand nombre de personne émettant au même moment dans le canal nous réussissons à récupérer le message qui nous concerne uniquement.

FIGURE 1 – affichage lors de l'exécution du programme