4AL05

Transmission d'informations sur le réseau électrique

Katerina Gorinskaia 3520807

Taki Soualhi 3872967

Codage de source

1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                            FONCTION ENTROPIE                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Cette fonction calcule les frequences d'apparition des
% symboles dans une source de message
% Elle calcule ensuite l'entropie de la source

function e = entropie(d)

% Conversion des données en entiers non signes (0 à 255)
data_int = uint8(d);

% Calcul des frequences d'apparition des lettres dans la phrase

freq = histc(data_int(:)', 0:255);  % Comptage des fréquences data_int dans une
intervalle 0-255

freq = freq(:)'/sum(freq);  % Vecteur avec les probabilités d’appariations de toutes
les fréquences
indice_lettres = find(freq);  % Suppression des entiers non présents, on stocke les
indices des probabilités non nuls
freq = freq(indice_lettres); % Les probabilités correspondantes aux fréquences
d'appariation

e=0;    % Initialisation de la variable e

% Calcul de l'entropie de la source
for i=1:length(freq)
    e=e+(freq(i)*log2(1/freq(i)));  % Formule de l'entropie, cf cours
end
```

3. voir code

Pour les calculs on a utilisé le fait que chaque lettre a été code sur 8 bits (code ASCII).

```
Le message suivant va etre compresse (par Huffman), transmis, puis
TESTDETI

L'entropie de la source est :
H=2.1556 bits

Le code calcule est :
D-->0001
E-->01
I-->0000
S-->001
T-->1

La longueur moyenne des symboles de code est :
n=2.8 bits

Sans compression, le nombre de symboles binaires par lettre source
n=8 bits

Le taux de compression est :
T=65 %
K>>
```

```
Le message suivant va etre compresse (par Huffman), transmis, puis
ABCDEFGHIJKLMNOPQRSTUVWXYZ

L'entropie de la source est :
H=4.7004 bits

Le code calcule est :
A-->10011
B-->10010
C-->10001
D-->10000
E-->01111
F-->01110
G-->01101
H-->01100
I-->01011
J-->01010
K-->01001
L-->01000
M-->00111
N-->00110
O-->00101
P-->00100
Q-->00011
R-->00010
S-->00001
T-->00000
U-->1111
V-->1110
W-->1101
X-->1100
Y-->1011
Z-->1010

La longueur moyenne des symboles de code est :
n=4.7692 bits

Sans compression, le nombre de symboles binaires par lettre source
n=8 bits

Le taux de compression est :
T=40.3846 %
K>>
```

```
Le message suivant va etre compresse (par Huffman), transmis, puis
CECIESTUNEDEMOCETTEPHRASEVAETRECOMPRESSEEPARLECODEDEHUFFMAN

L'entropie de la source est :
H=3.6923 bits

Le code calcule est :
A-->0110
C-->0000
D-->1111
E-->10
F-->01110
H-->01011
I-->010101
L-->010100
M-->1110
N-->01001
O-->1101
P-->1100
R-->0011
S-->0010
T-->0001
U-->01000
V-->01111

La longueur moyenne des symboles de code est :
n=4.4118 bits

Sans compression, le nombre de symboles binaires par lettre source
n=8 bits

Le taux de compression est :
T=44.8529 %
K>>
```

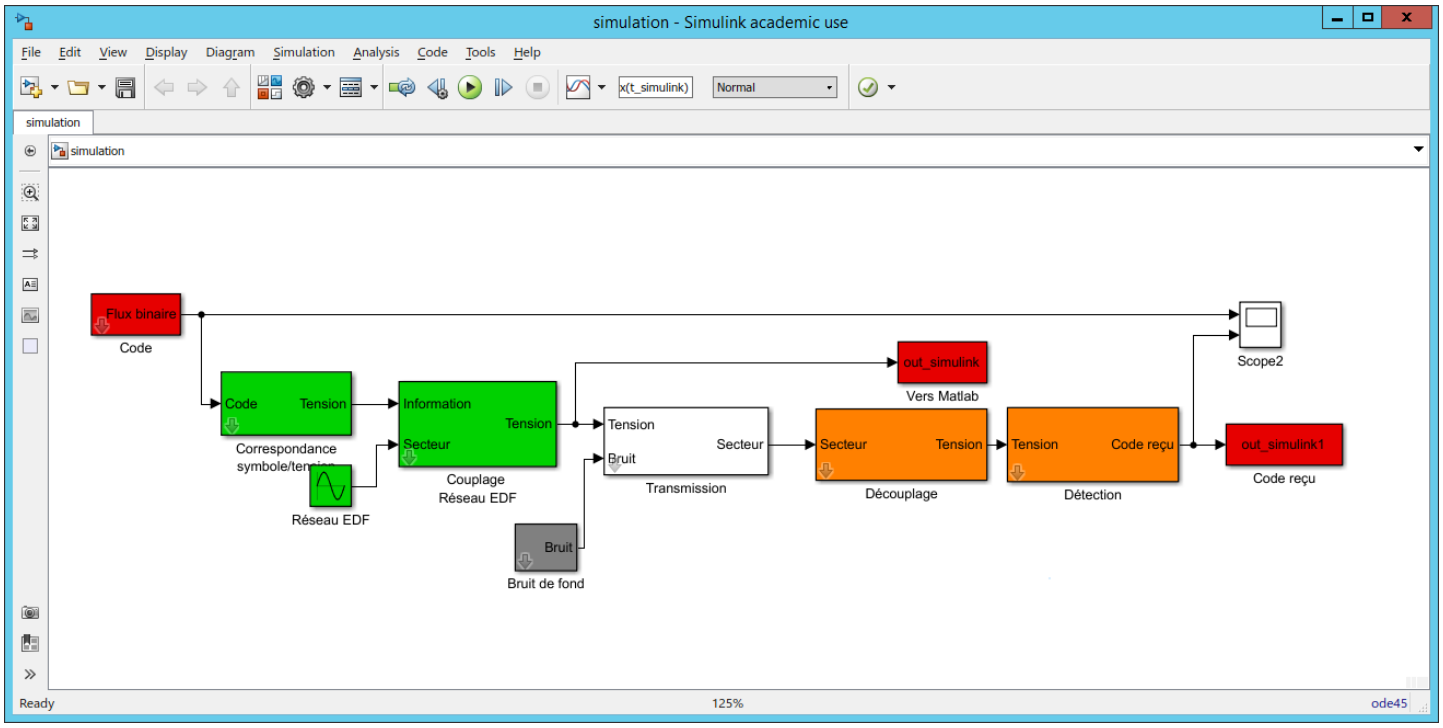
Pour les 3 textes données, la meilleur compression était obtenu pour la chaine ‘TESTDETI’ tandis que la mauvaise compression était obtenu pour l’ensemble des lettres alphabétiques. C’est à cause de l’entropie qui était plus petite dans le premier cas par rapport aux autres. Plus l’entropie est grande plus le désordre dans l’information est existant, donc la compression sera moins efficace.

Codage de canal

- voir code

Transmission

Schéma complet Simulink :



Codage en bande de base

6. Le réseau EDF envoie le signal électrique avec la fréquence 50 Hz (standard pour les réseaux en Europe). On peut voir l’harmonique principale à 50 Hz sur l’allure fréquentielle.
Le graphe est symétrique autour de $f_e/2$ = 50 kHz/2 = 25 kHz. On a choisi f_e de 50 kHz grâce à la théorème de Shannon (fréquence porteuse = 10kHz, 50 kHz > 10 kHz*2)

Le reste du graphe correspond à l’information utile à transmettre + bruit.

On observe aussi un pic autour 10 kHz – fréquence porteuse, beaucoup plus élevée que celle du signal d'entrée.

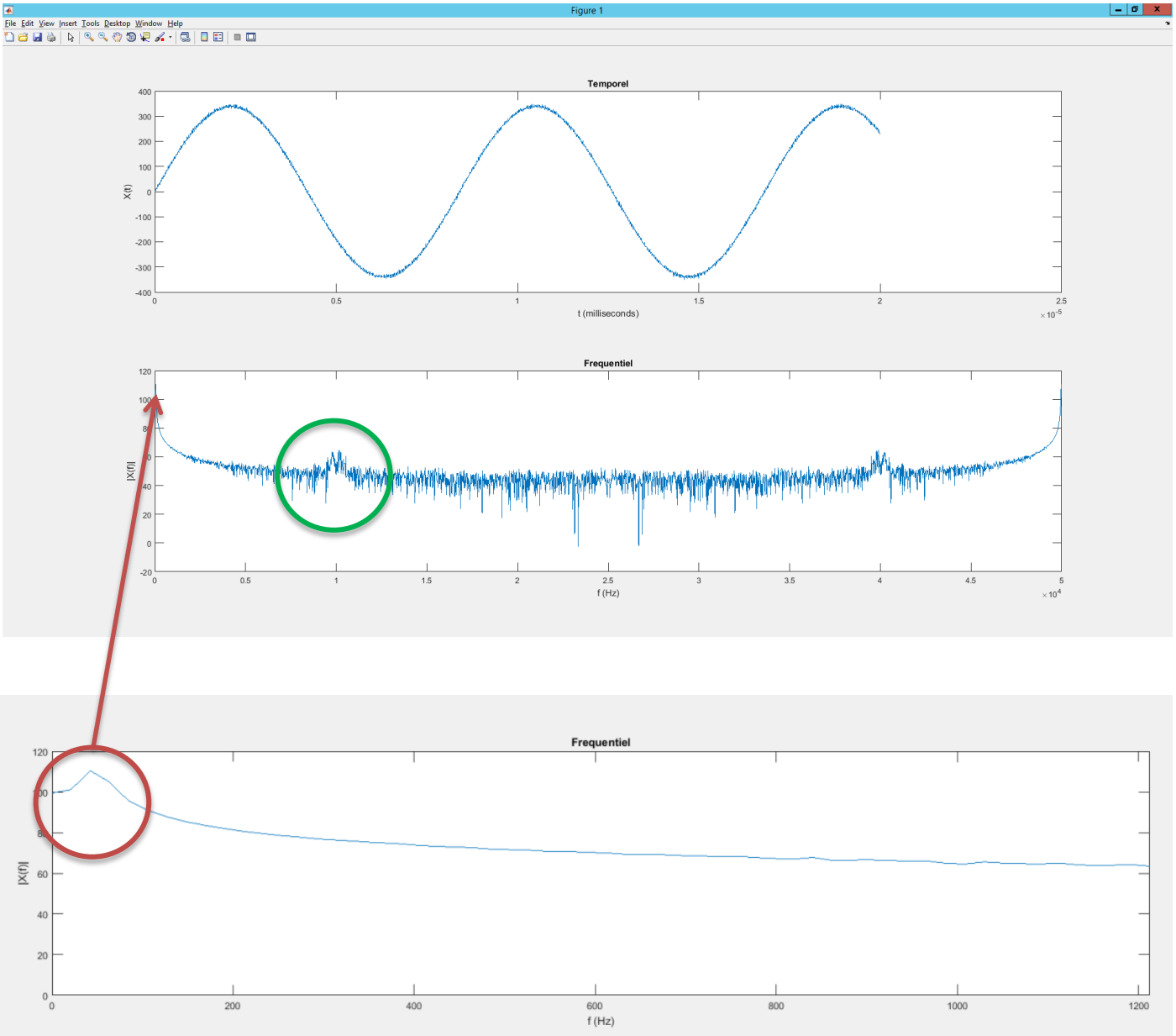
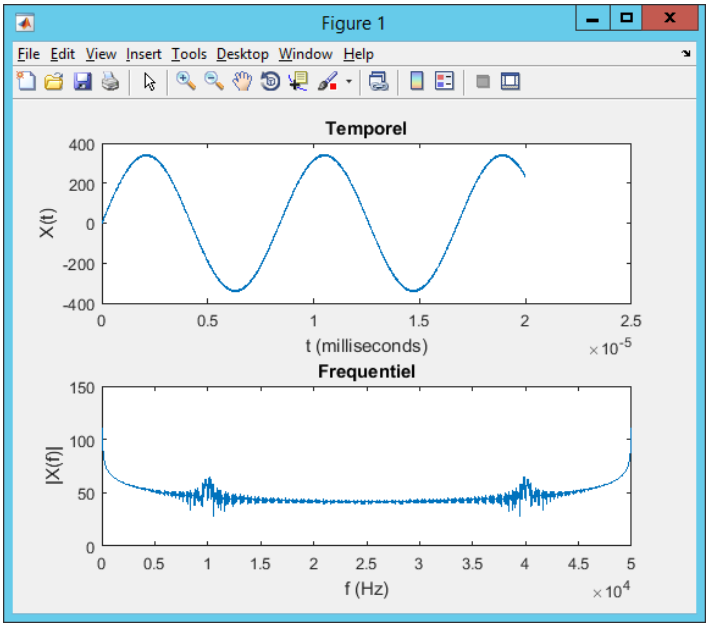


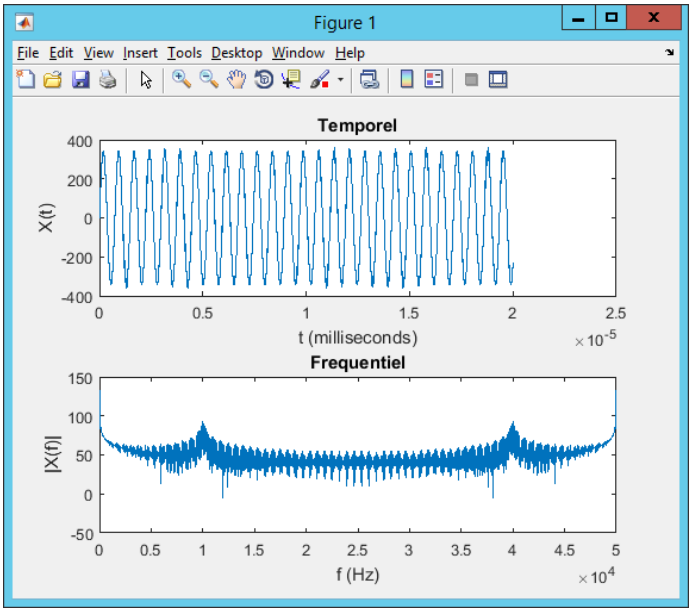
Figure 1
File Edit View Insert Tools Desktop Window Help
Temporel
X(t)
t (milliseconds)
Frequentiel
|X(f)|
f (Hz)

Frequentiel
|X(f)|
f (Hz)

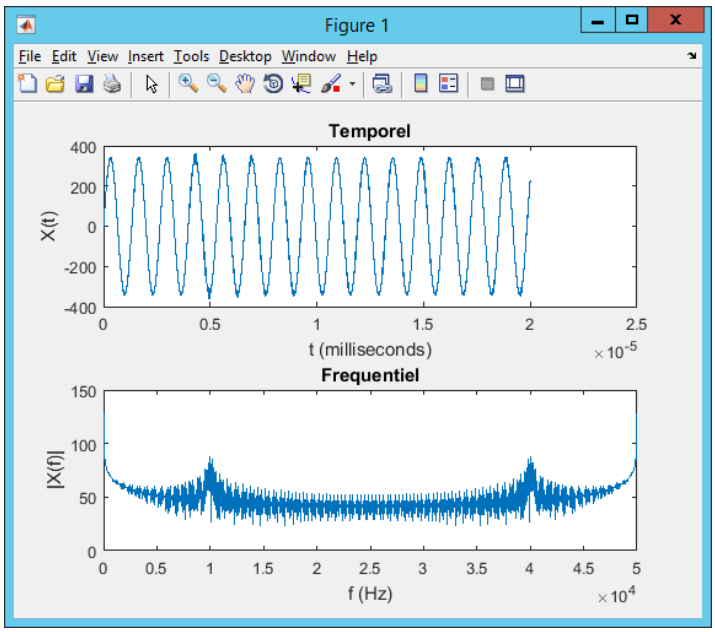
7.



Texte 1



Texte 2



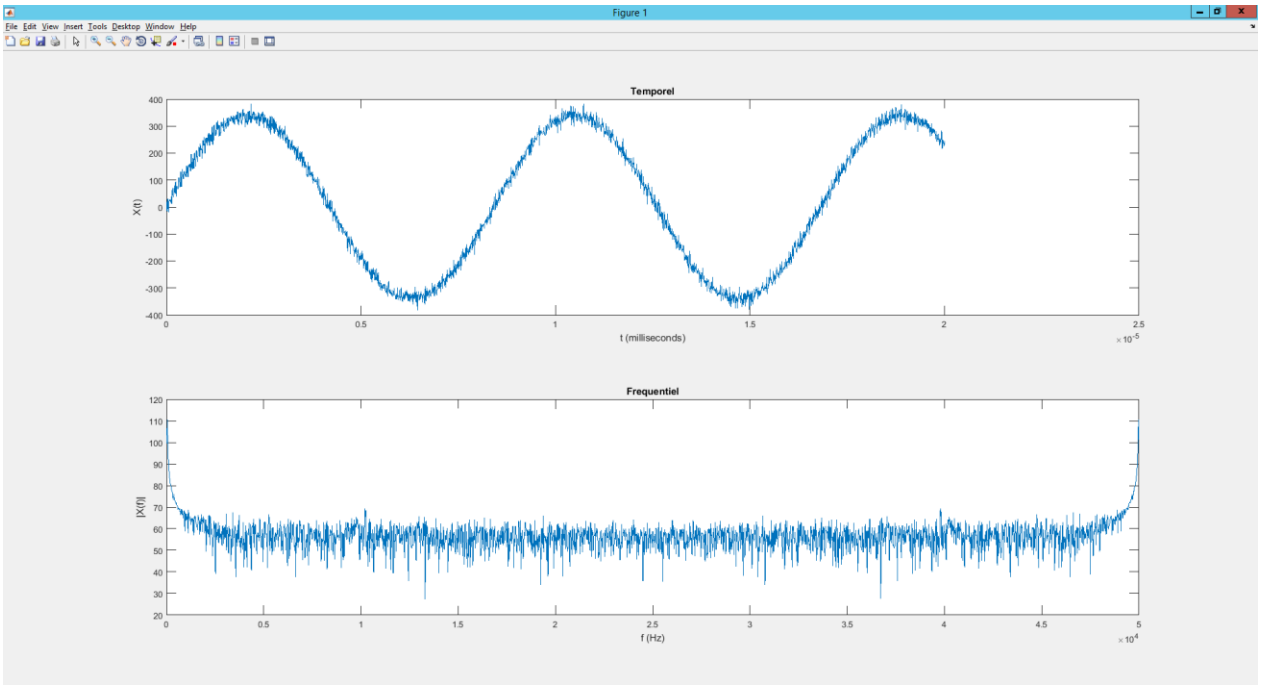
Texte 3

Dans notre cas, pour un texte 1 court, un texte 3 moyen et un texte 2 long, on remarque que le texte 1 a une occupation spectrale minimale par rapport aux autres, et le texte 2 la maximale.

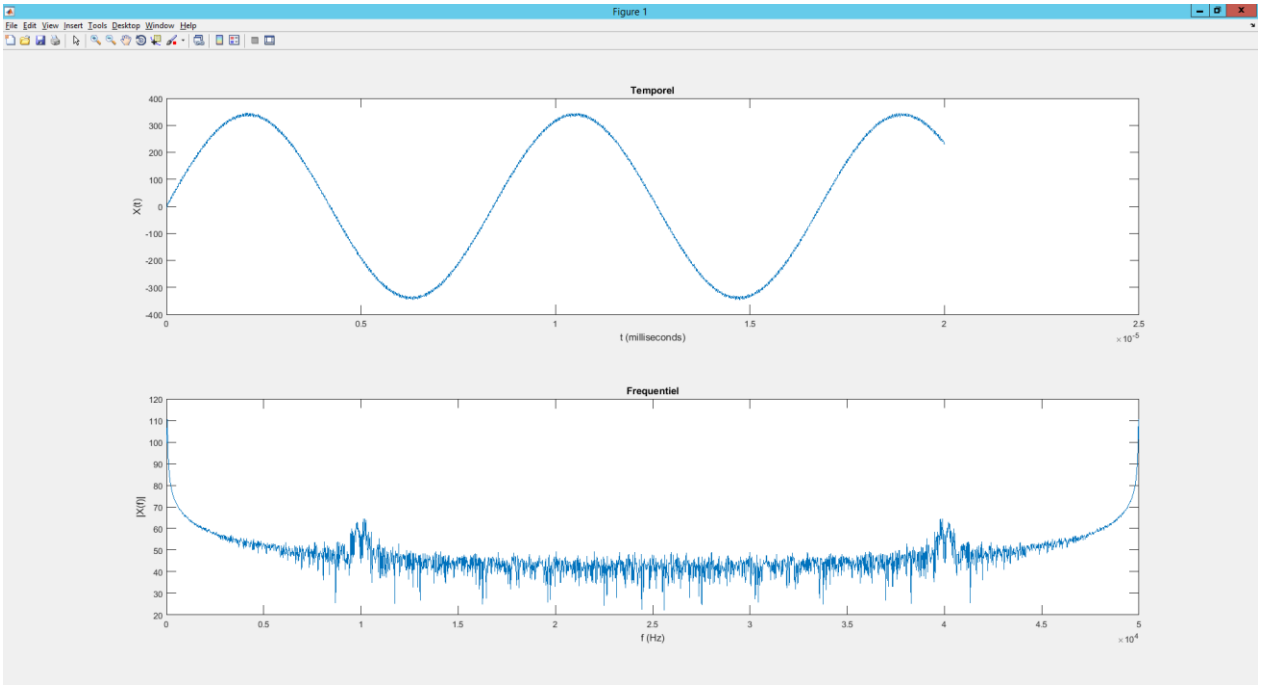
Donc, plus la taille de l'information est grande plus son spectre est large.

9. En changeant la valeur de RSB, on remarque visuellement, qu'avec les valeurs négatives la présence du bruit est très importante. Plus RSB est grande, moins du bruis est présente sur le graphe.

$$S/B_{dB} = 10 \cdot \log(S/B)$$



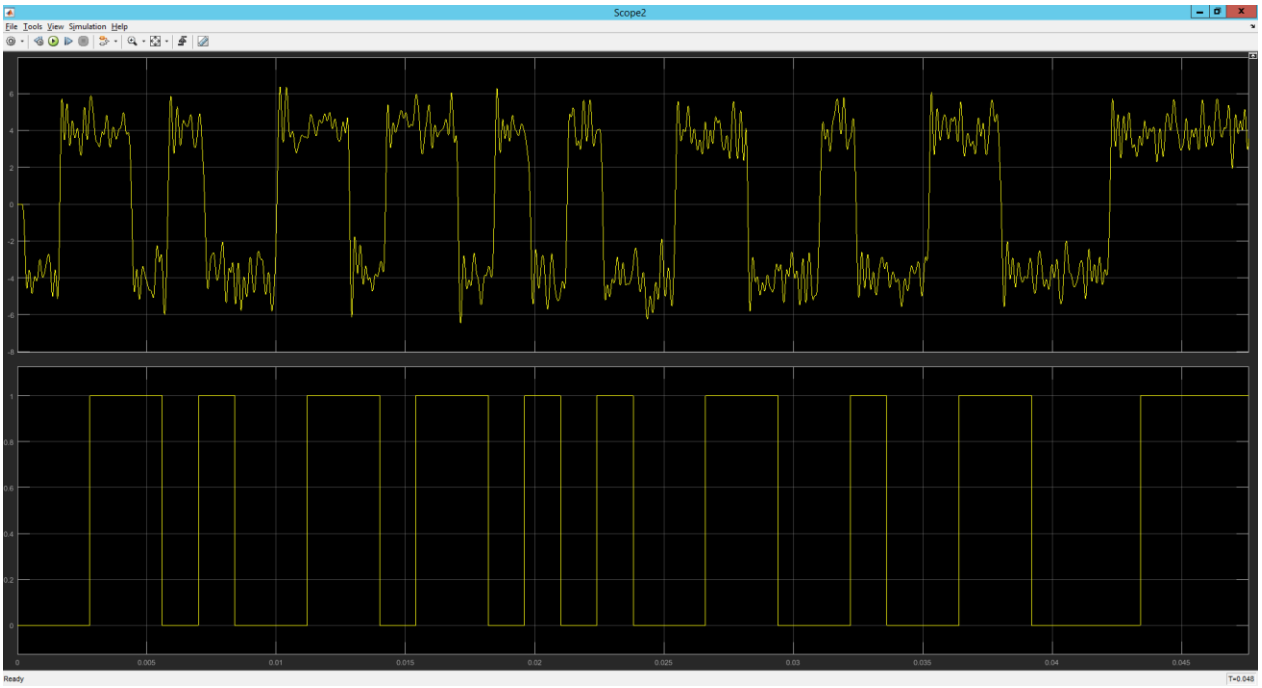
RSB = -10



RSB = 10

Découplage : ce bloc sert à extraire l'information superposée au courant.

Détection : c'est un bloc comparateur avec la valeur de seuil à 0, il permet d'obtenir le signal numérique à partir du signal analogique.

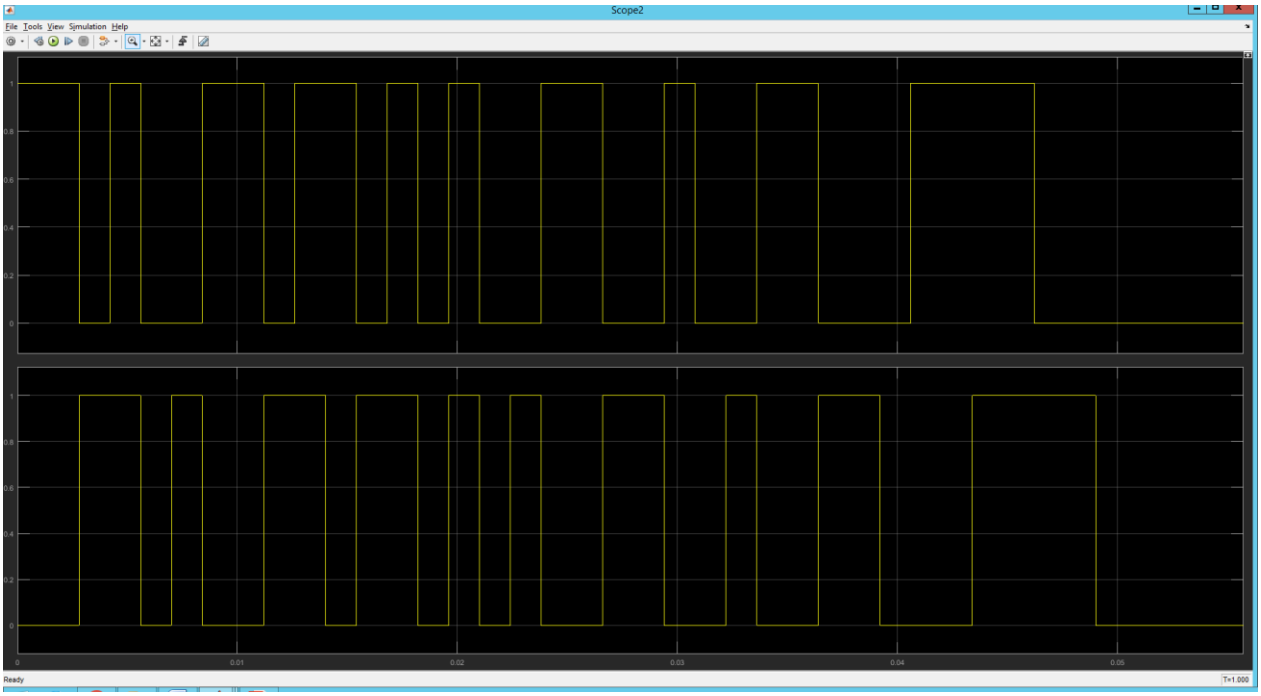


En utilisant un scope, on visualise les sorties des deux block Découplage et Détection. On peut confirmer notre hypothèse, c'est évident que le bloc Détection sert à convertir le signal en entré a un signal numérique. De plus, il rajout un décalage sur l'axe du temps.

Décodage

2. On constate que avec RSB infinie, l'information émis 'code' correspond à l'information reçu '*out_simulink1'*, mais elle est décale dans le temps avec un certain nombre de bits.

3.



Sur le scope, on a calculé 18 bits a ‘1’ et ‘17’ bits a ‘0’ , donc le taux d’erreur de transmission de l’information est nul. De plus, on peut remarquer la décalage de deux bits entre le code emis et le code reçu.

5.
RSB = -5 dB

```
Taux d'erreur caractère (SANS code correcteur):
  - Nombre d'erreur : 6
  - Taux d'erreur : 17.14%
Taux d'erreur caractère (AVEC code correcteur) :
  - Nombre d'erreur : 5
  - Taux d'erreur : 14.29%
>>
```

10. On peut conclure, que le code correcteur n’est pas suffisant pour corriger toutes les erreurs de transmission pour le bruit élevée.

Cryptage des données

1. a) Chiffrement de Vigenère – algorithme symétrique avec une clé privée. Avec une clé constituée d'un seul caractère cela correspond à un **code de Caesar**.

c) On regarde la fréquence d’appariation des lettres différentes et on choisi la lettre la plus fréquente. En Français, la lettre la plus couramment utilisée est **e**. Donc, par correspondance, on peur facilement déduire la clé. Ca marche dans le cas si le code chiffré - c’est le texte en français.

3. Dans le cas ou le bruit est négligeable ou le RSB est grand, le correcteur est capable de corriger une erreur si elle est présente, donc le texte émis sera le même que le texte reçu.

Par contre si le RSB est trop petit, donc un bruit considérable existe par rapport à l’information, la possibilité d’avoir des erreurs augmente et le correcteur utilisé n’est pas suffisant pour les corriger tout, donc le texte émis et le texte reçu ne seront pas identiques.

Evolution des taux d'erreurs en fonction du RSB

Pour la chaine de caractère suivante : TESTDETI

On a calculé le taux d’erreur pour différentes valeurs de RSB et on a tracé le graphe de correspondance. On peur bien voire la dépendance linéaire sur un certain intervalle et à partir d’une certaine valeur le taux d’erreur est constant (= 0). Cela signifie que ce n’est pas possible d’avoir plus qu’un certain nombre des erreurs corrige, dans notre cas le taux d’erreurs devient 0, on n’as plus des erreurs.