Rapport Matlab : Simulation d'une chaîne de transmission numérique sur canal gaussien à bande limitée

Hoël Boëdec
ENSIMAG - ISSC
3 rue Amiral Courbet
Grenoble, France
hoel.boedec@phelma.grenoble-inp.fr

Fournier Mickaël
ENSIMAG - ISSC
22 rue Francis Jaquard
Grenoble, France
mickael.fournier@phelma.grenoble-inp.fr

1. INTRODUCTION

2. GÉNÉRATION ALÉATOIRE DES ÉLÉMENTS BINAIRES

```
N = 2048;

bn = zeros(1, N);
for k=1:1:length(bn)
    bn(k) = round(rand());
end

mean(bn);
var(bn);
```

La moyenne et la variance empirique de bn valent respectivement 0,5 et 0,25. Ceci est cohérent avec la théorie car 0 et 1 sont équiprobables et indépendants.

3. CONVERSION DES ÉLÉMENTS BINAIRES EN SYMBOLES (MAPPING)

```
an = zeros(1, N);
for k=1:1:length(bn)
    an(k) = 2*bn(k)-1;
end

mean(an);
var(an);

mean(an.^2);
```

La moyenne et la variance empirique de an valent respectivement 0 et 1. Ceci est cohérent avec la théorie car les symboles sont centrés, équiprobables et indépendants. La puissance moyenne temporelle empirique du vecteur an vaut 1 (unité ??).

```
N = 32;

D = 10000000; # 1 Mbit/sec

T = 1/D;

t_a = 0 : T : N*T - T;

plot(t_a, an, '*')
```

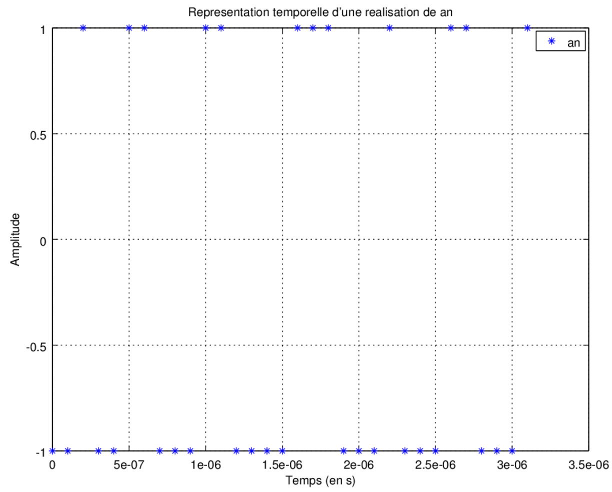

Figure 1: Représentation temporelle d'une réalisation de an

```
plot(real(an), imag(an), '*')
```

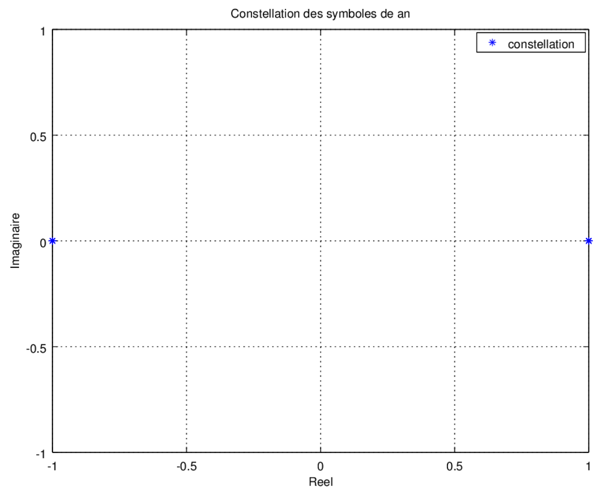

Figure 2: Constellation des symboles de an

4. CONVERSION NUMÉRIQUE - ANALOGIQU

4.1 Expansion - Question 1

La durée du signal st peut s'écrire : NF/D.

```
N = 32;

F = 16;
st = zeros(1, N*F);
st(1) = F*an(1);
for k=1:1:length(an)-1
    st(k*F+1) = F*an(k+1);
end
```

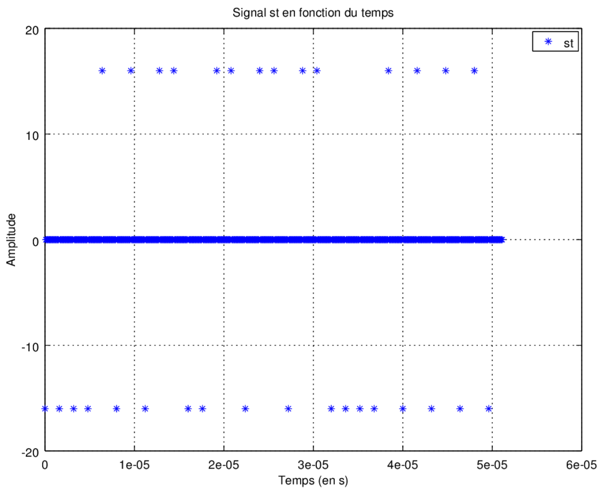

Figure 3: Signal st en fonction du temps

4.2 Etude des filtres - Question 2

```
N = 32;

L = 8;
alpha = 0.5;
Te = T/F;
t_filtre = [0 : T/F : L*T -T/F];
```

4.2.1 NRZ

```
s_t = gen_filters2('nrz',t_filtre,T,F,L,alpha);
plot(t_filtre, s_t, '*')
```

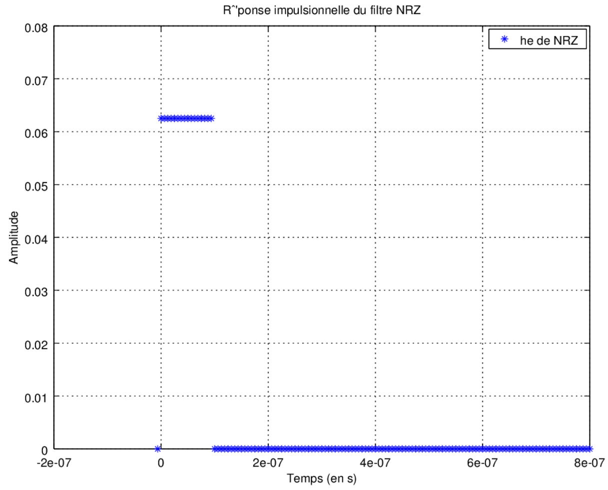

Figure 4: Réponse impulsionnelle du filtre NRZ

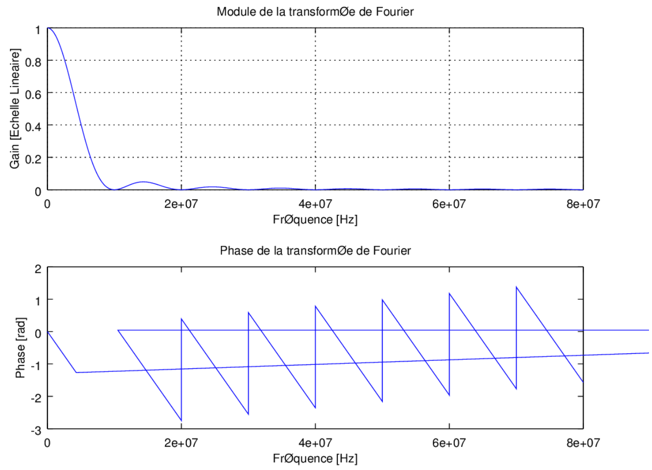

Figure 5: Module de la transformée de Fourier de la réponse impulsionnelle du filtre NRZ

4.2.2 RZ

```
s_t = gen_filters2('rz',t_filtre,T,F,L,alpha);
plot(t_filtre, s_t, '*')
```

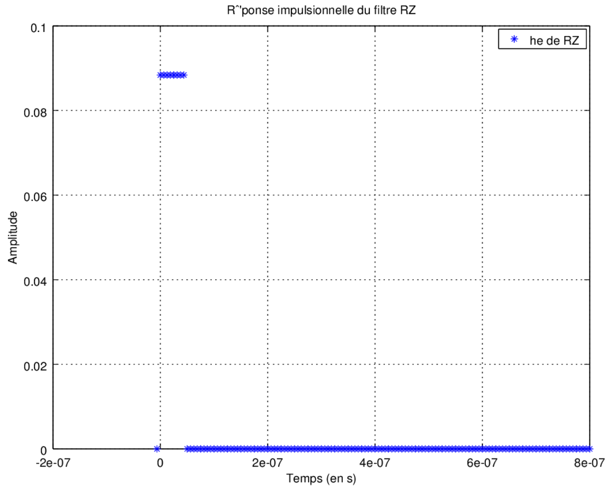

Figure 6: Réponse impulsionnelle du filtre RZ

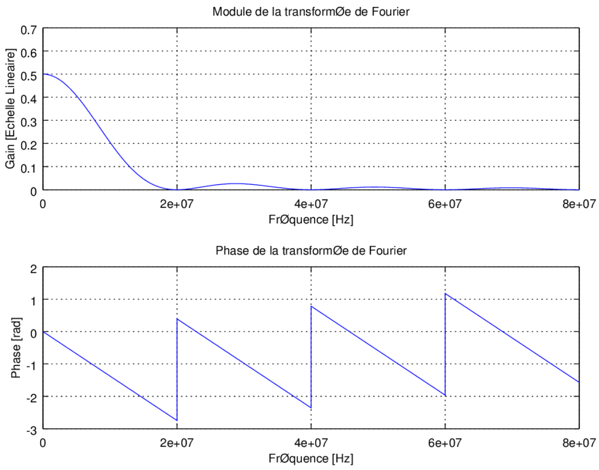

Figure 7: Module de la transformée de Fourier de la réponse impulsionnelle du filtre RZ

4.2.3 SRRC

```
s_t = gen_filters2('srrc',t_filtre,T,F,L,alpha);
plot(t_filtre, s_t, '*')
```

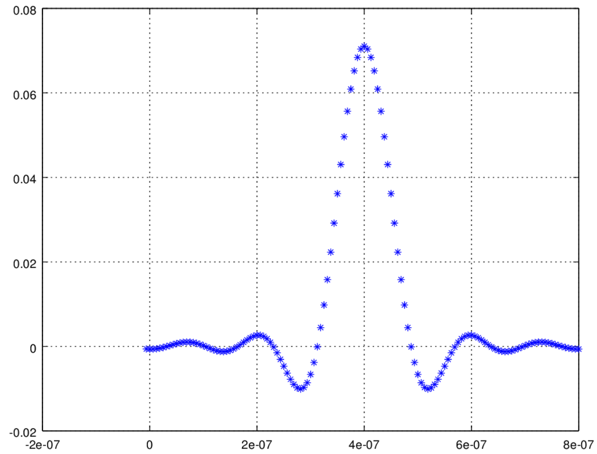

Figure 8: Réponse impulsionnelle du filtre SRRC

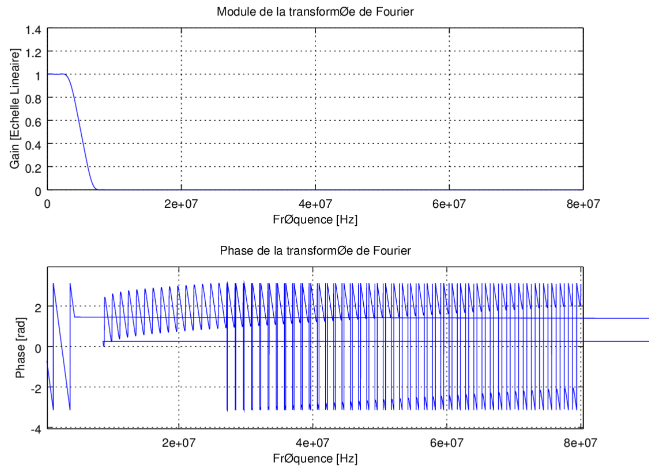

Figure 9: Module de la transformée de Fourier de la réponse impulsionnelle du filtre SRRC

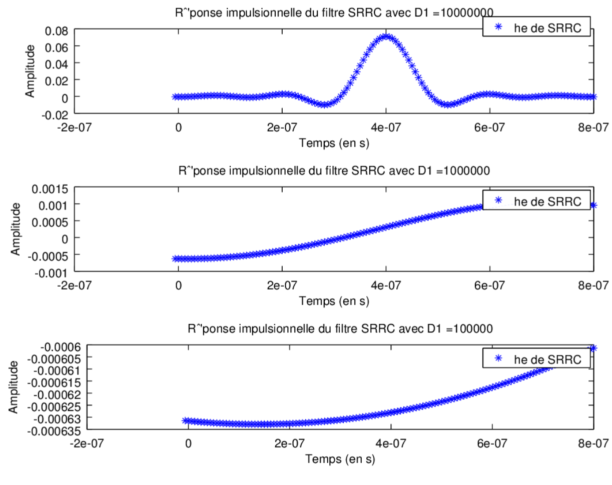

Figure 10: Variation de la réponse impulsionnelle du filtre SRRC avec le débit

Interêt de D :

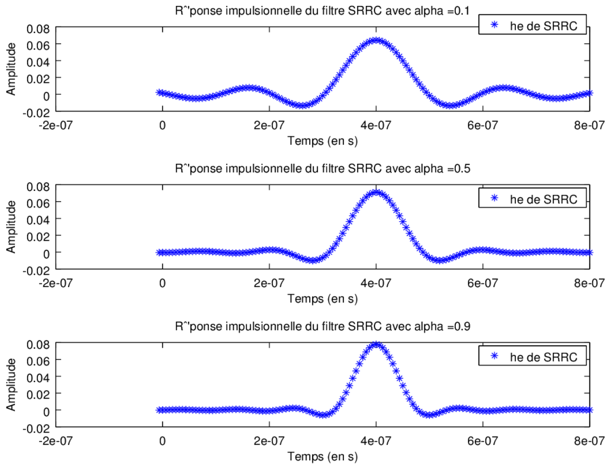

Figure 11: Variation de la réponse impulsionnelle du filtre SRRC avec alpha

Interêt de alpha :

Est-ce logique d'observer une variation de phase linéaire avec la fréquence ?

4.3 Mise en forme des symboles

4.3.1 Question 3

```
N = 32;

t_x = -L*F*T/2 : T : (N*F)*T + L*F*T/2;
ht = gen_filters2('srrc',t_filtre,T,F,L,0.5);
xt = conv(st, ht);
figure; subplot(2,1,1); plot(t_s, st, '*');
subplot(2,1,2); plot(t_x, xt);
```

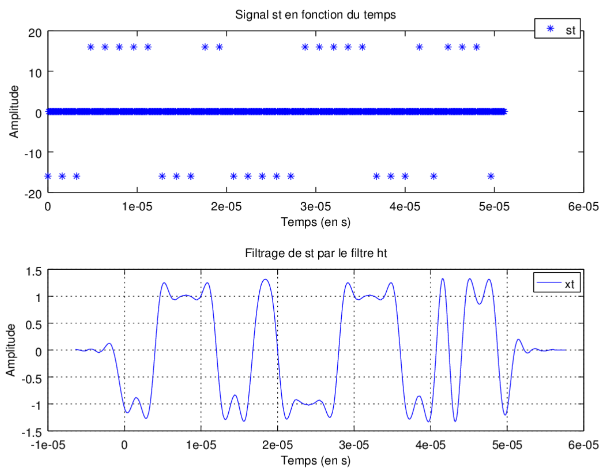

Figure 12: Signal st et son filtrage par ht en fonction du temps

```
length(st)
length(ht)
length(t_x)*T
length(t_filtre)*T
```

Résultats : ans = 512 (16*32) ans = 130 (LF) ans = 6.4100e-05 (cf ts) ans = 1.3000e-05 (cf tfiltre)

4.3.2 Question 4

```
N = 2048;

mean(xt.^2)
```

La puissance moyenne empirique du signal émis xt vaut 1. Ceci cohérent avec la théorie car la variance vaut 1 et le norme carrée du filtre d'émission vaut 1/T (filtre normalisé).

4.3.3 Question 5

4.3.4 Question 6

5. AJOUT DU BRUIT BLANC GAUSSIEN - QUESTION 7

On sait que $sigman^2 = \frac{NoF}{2T}$ or P(xt) = $\frac{Eb}{T}$ d'où la formule $\frac{Eb}{N0} = \frac{F}{2} * (\frac{P(xt)}{sigman^2})$.

De plus, on a P(xt) = 1 ici, donc sigman² = f($\frac{Eb}{N0}$) avec f(x) = $\frac{F}{2x}$.

```
EbN0 = 100;

sigma_n = sqrt((F/2)/EbN0);
nt=sigma_n*randn(1,length(xt));
rt = xt + nt;
```

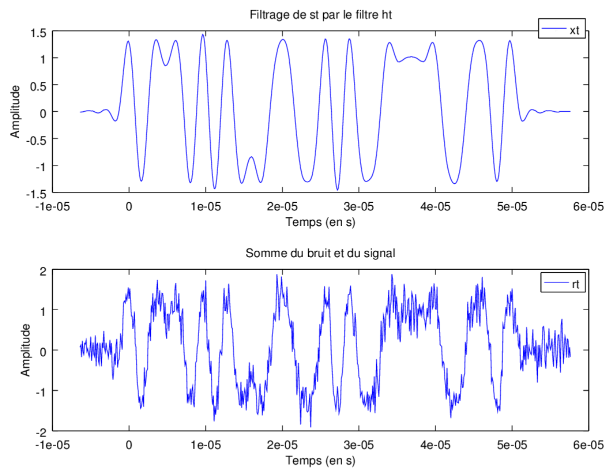

Figure 13: Filtrage de st par le filtre ht et le même signal bruité

6. CONVERSION ANALOGIQUE - NUMÉRIQU

6.1 Filtrage adapté

6.1.1 Question 8

Définition de filtre adapté :
Justification utilisation filtre adapté dans chaîne de communication :

6.1.2 Question 9

```
N = 32;

htr = fliplr(ht+L*T);
plot(t_filtre, ht, '*');
plot(t_filtre, htr, 'r+');
```

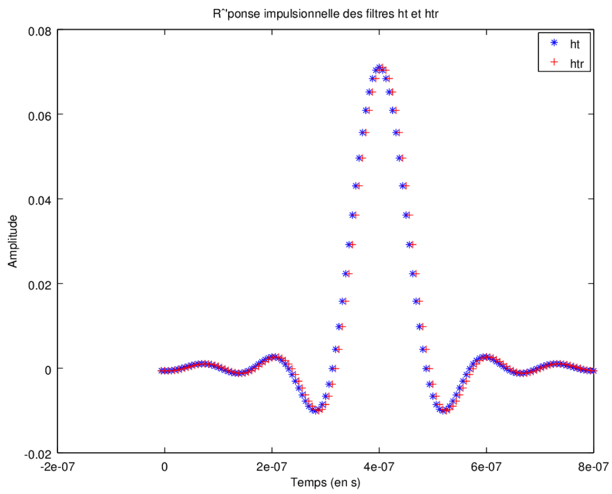

Figure 14: Réponse impulsionnelle des filtres ht et htr

Justifier la forme de la réponse impulsionnelle du filtre adapté ht(r) :

```
pt = conv(ht, htr);

t_filtre_pt = [0 : T/F : 2*(L*T + T/F)];
figure; plot(t_filtre, ht, '*'); hold on;
plot(t_filtre_pt, pt, 'r+');
```

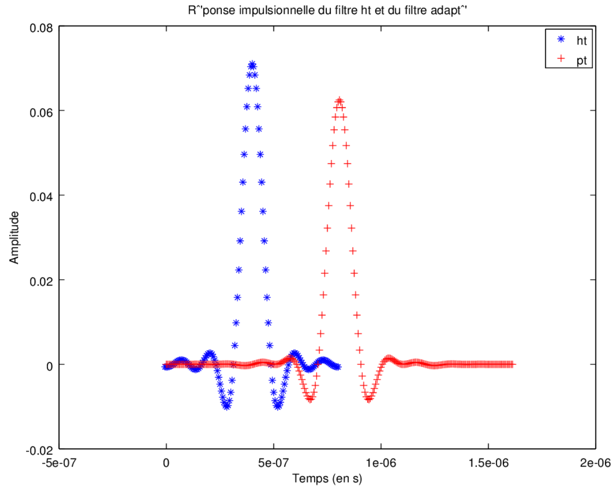

Figure 15: Réponse impulsionnelle du filtre ht et du filtre adapté

Pt est construit comme le retourné décalé de ht. Ici on a choisi LT comme décalage afin que le filtre équivalent émission-réception soit causal.
On sait que le vecteur temps t_filtre_pt vaut le double du vecteur t_filtre par convolution.

6.1.3 Question 10

A l'aide de Matlab on fait crée un vecteurs contenant les valeurs de pt en to + nTs avec to = L*T et Ts = T.

```
vect_result = zeros(1,17);

for k=1:1:8
  vect_result(9-k) = pt(round((L*T-k*T)/(T/F)));
  vect_result(9+k) = pt(round((L*T+k*T)/(T/F)));
end

vect_result(9) = pt(round((L*T)/(T/F)));

t_vect_result = [L*T - 8*T: T : L*T + 8*T];
plot(t_vect_result, vect_result, '*')
```

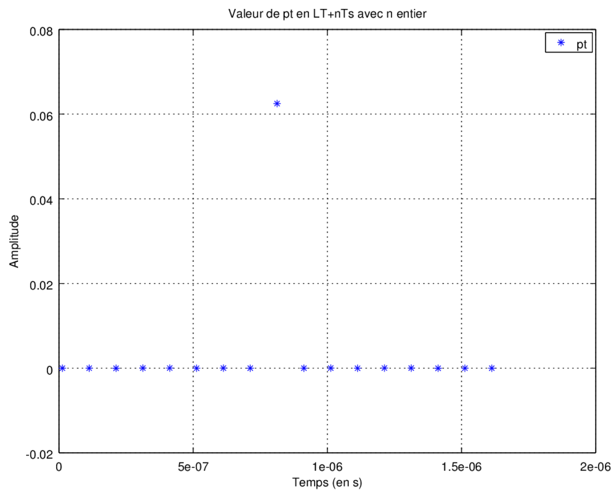

Figure 16: Valeur de pt en LT+nTs avec n entier

On voir que pour n <> 0 les valeurs de pt sont bien nulles. On ne peut pas transmettre sans IES car on ne peut atteindre une précision telle que pt aura une valeur nulle tous les to + nTs. Cependant, on peut s'en approcher, et donc minimiser l'IES, en ???.

Diagramme de l'oeil : j'ai un truc mais .. ça m'a pas l'air d'être ça du tout

6.1.4 *Question 11*

blablablablabla

6.2 Décimation

6.2.1 *Question 12*

cf cours

6.2.2 *Question 13*

```
t_y = -L*F*T : T : (N*F)*T + L*F*T + T;
yt = conv(rt, htr);

plot(t_x, xt); hold on; plot(t_y, yt, 'r');
```

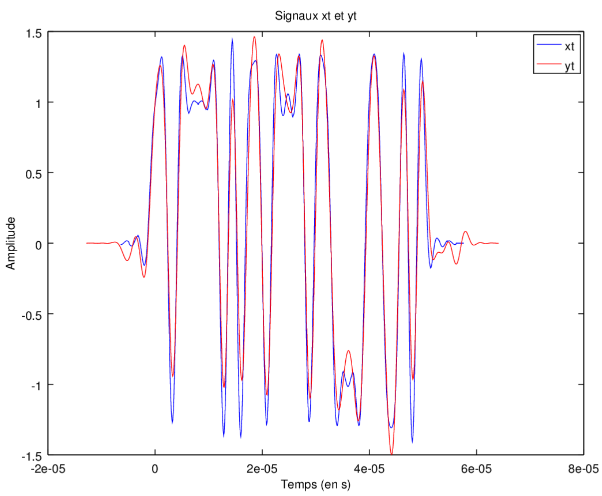

Figure 17: Signauc xt et yt

```
yk = [];

for k=1:1:length(yt)-1
    if mod(k, F) == 0
      && -L*F*T + k*T >= 0
        && -L*F*T + k*T < (N*F)*T
        yk = [yk F*yt(k)];
    end
end

t_yk = [0 : F*T : (N*F-1)*T];
figure; plot(t_s, st, '*'); hold on;
plot(t_yk, yk, 'r+')
```

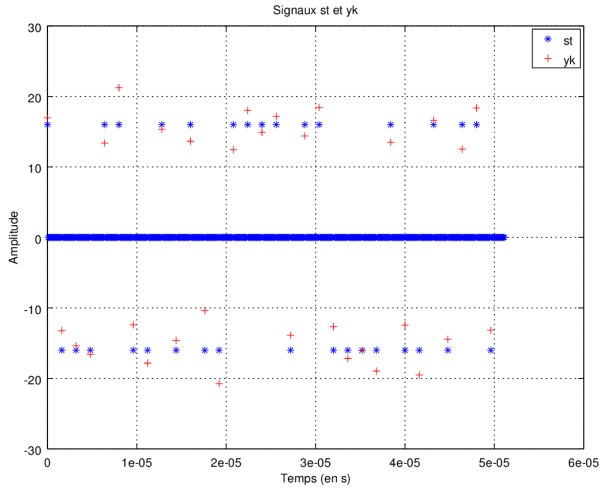

Figure 18: Signaux st et yk

On vérifie grâce à la fonction length que le vecteur yk obtenu est bien de la même taille que an, soit 32.

7. PRISE DE DÉCISION (DEMAPPING)

On choisit comme seuil sigmaN (rapport avec le bruit)

```
bnF = [];

for k=1:1:length(yk)
    val = yk(k)/16;
    if val > 1 - 2*sigma_n
        bnF = [bnF 1];
    else
        bnF = [bnF 0];
    end
end

figure; plot(t_a, bn, '*'); hold on;
plot(t_a, bnF, 'ro');
```

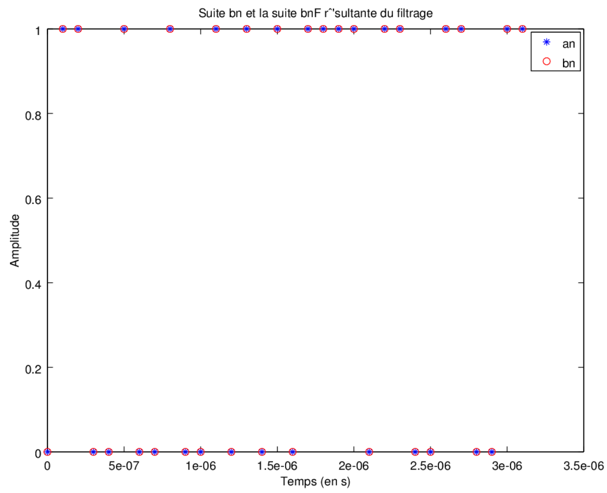

Figure 19: Signaux bn et la suite bnF résultante du filtrage

8. CALCUL DU TAUX D'ERREUR BINAIRE

```
sum(xor(bn, bnF))/N
```

On trouve 0.027958 (avec N = 262144)

9. MESURES DE PERFORMANCES

```
val_ebno = [50:200:20000];

vect_val_pratique = [];

for k=1:1:length(val_ebno)
    vect_val_pratique =
      [vect_val_pratique teb(2048, val_ebno(k))];
end

plot(val_ebno, vect_val_pratique);
```

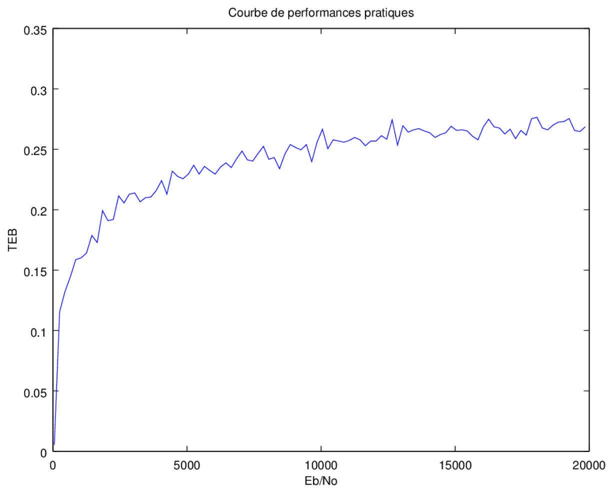

Figure 20: Courbe de performances pratiques

10. OPTIONNEL

10.1 Autres impulsions de mise en forme

10.2 Rapport signal à bruit sur la variable de décision

10.3 Analyseur de spectre

11. CONCLUSION