Jeanniard Sébastien

Lemaître Guillaume

TP n°2 : Analyse et traitement de signal sous Matlab

2.4 Production de signaux sous Matlab :

2.4.1 Synthèse d'un signal sinusoïdal :

2.4.1.1 Théorème de Shannon :

Le programme permettant de généré un signal sinusoïdal est le suivant :

```
Ts=100;                 % Période du sinus
Duree=200;              % Durée d'observation signal 2 périodes
dt=1;                   % Pas ou période d'échantillonnage temporel du signal
N=Duree/dt;             % Nombre total d'échantillons
n=0:N;                  % "Vectorisation du temps" (échantillonnage)
t=n*dt;                 % Définition du temps
y=sin(2*pi*t/Ts);       % Génération du sinus
```

Dans les manipulations qui suivent, nous allons uniquement modifier la valeur de la variable dt qui re présente la période d'échantillonnage.

- Cas où dt = 1 :

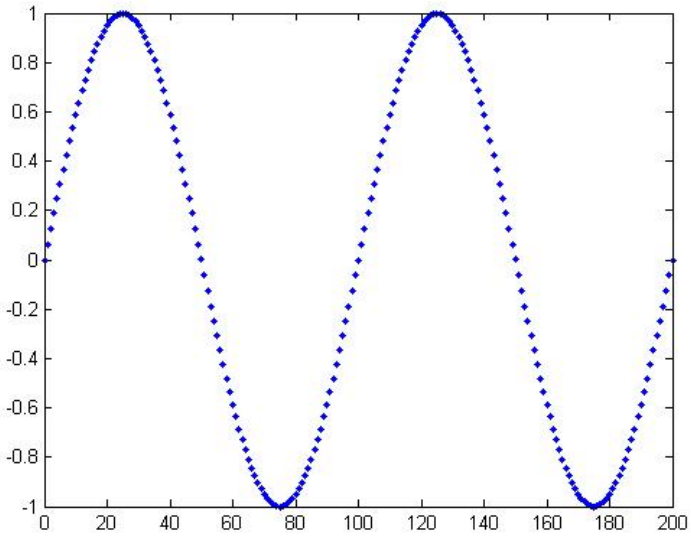


Figure 1 : Cas où dt = 1

Dans ce cas nous pouvons constater que la fréquence d'échantillonnage est largement suffisante.

Nous pouvons préciser que la fréquence d'échantillonnage vaut :

$$F_e = 1\ Hz$$

- Cas où dt = 10 :

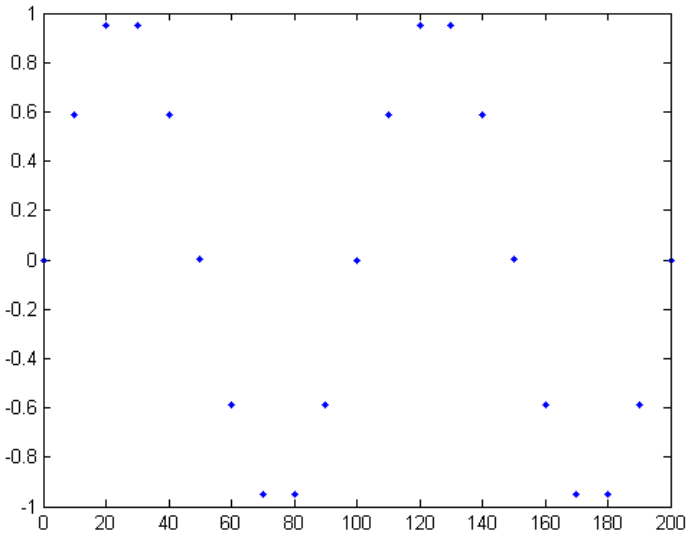


Figure 2 : Cas où dt = 10

Nous pouvons constater que la fréquence d'échantillonnage est toujours suffisante pour synthétiser un signal sinusoïdal.

Nous pouvons préciser que la fréquence d'échantillonnage vaut :

$$F_e = \frac{1}{10} = 0{,}1\ Hz$$

- Cas où dt = 50 :

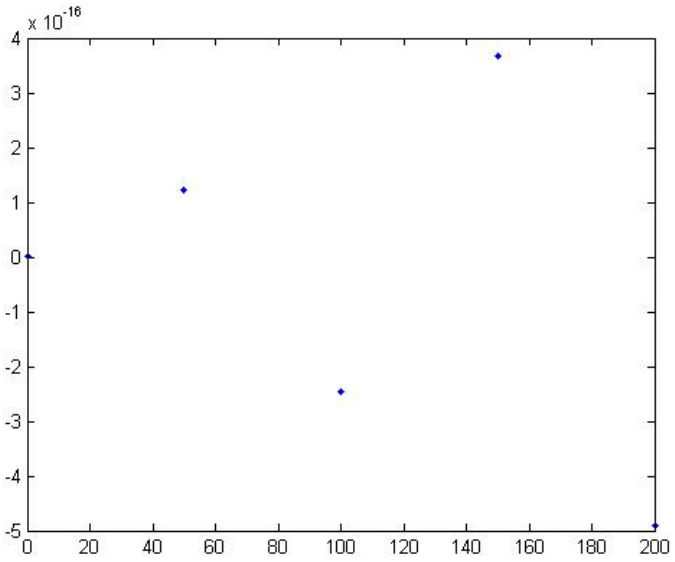


Figure 3 : Cas où dt = 50

Ce cas est un le cas de sous-échantillonnage où la fréquence d'échantillonnage est trop faible.

Nous pouvons préciser que la fréquence d'échantillonnage vaut :

$$F_e = \frac{1}{50} = 0{,}02\ Hz$$

Nous pouvons recouper ces résultats en introduisant le théorème de Shannon qui nous indique que la fréquence d'échantillonnage doit être supérieure à deux fois la fréquence maximale du signal.

Dans notre cas, la fréquence du signal sinusoïdal est de :

$$f_{max} = \frac{1}{100} = 0{,}01\ Hz$$

D'où en théorie, nous devons avoir une fréquence d'échantillonnage tel que :

$$F_e > 2 \times f_{max}$$

$$F_e > 0{,}02\ Hz$$

Les résultats précédemment trouvé expérimentalement suivent bien ce théorème.

2.4.1.2 Ajout d'un bruit Gaussien :

Le programme pour ajouter un bruit Gaussien de variance σ = 0.3 :

```
Ts=100;                 % Période du sinus
Duree=200;              % Durée d'observation signal 2 périodes
dt=1;                   % Pas ou période d'échantillonnage temporel du signal
N=Duree/dt;             % Nombre total d'échantillons
n=0:N;                  % "Vectorisation du temps" (échantillonnage)
t=n*dt;                 % Définition du temps
```

```
y=sin(2*pi*t/Ts);          % Génération du sinus
B=0.3;                     % B écart type de la distribution gaussienne du bruit
bruit=B*randn(1,N+1);          % Note: variance=B^2;
yb=y+bruit;                % Signal bruité
```

Nous obtenons le signal bruité suivant :

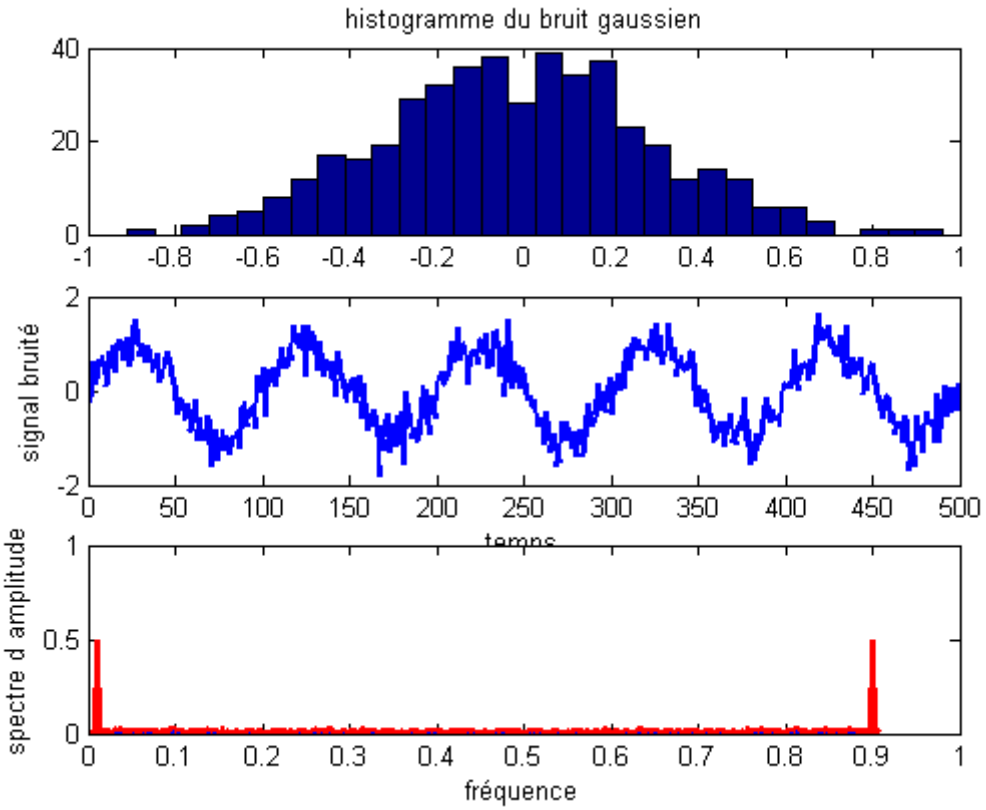


Figure 4 : Signal sinusoïdal bruité par un bruit Gausssien d'écart-type égal à 0,3

Le programme pour ajouter un bruit Gaussien de variance $\sigma^2 = 0.16$ c'est-à-dire une variance $\sigma = 0.4$

```
Ts=100;                    % Période du sinus
Duree=200;                 % Durée d'observation signal 2 périodes
dt=1;                      % Pas ou période d'échantillonnage temporel du signal
N=Duree/dt;                % Nombre total d'échantillons
n=0:N;                     % "Vectorisation du temps" (échantillonnage)
t=n*dt;                    % Définition du temps
y=sin(2*pi*t/Ts);          % Génération du sinus
B=0.4;                     % B écart type de la distribution gaussienne du bruit
bruit=B*randn(1,N+1);          % Note: variance=B^2;
yb=y+bruit;                % Signal bruité
```

Nous obtenons le signal bruité suivant :

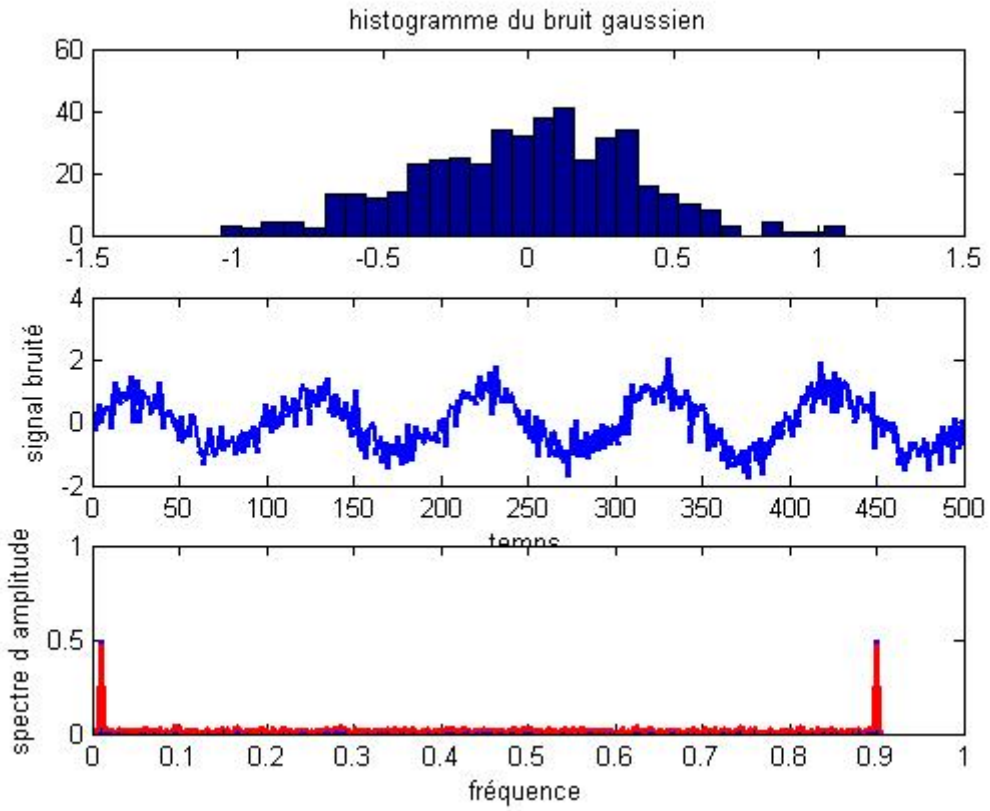


Figure 5 : Signal sinusoïdal bruité par un bruit Gaussien de variance égale 0.16

2.4.2 Synthèse de Fourier :

2.4.2.1 Cas d'un signal créneau :

Le programme permettant de calculer la décomposition en série de Fourier d'un signal créneau est le suivant :

```
n=input('Entrer le nombre de coefficient n pour la synthèse ');
dt=1./(50*n*f0);          % Incrément temporel
t=(0:dt:2/f0);            % Création du temps sur deux périodes du signal par incrément de dt
A0=A/2;                   % Valeur moyenne du signal
e=A0;                     % Boucle de calcul de la somme
for i=1:n
        an=A*sinc(i/2);
        bn=0;
        xn=an*cos(2*pi*f0*i*t)+bn*sin(2*pi*f0*i*t);
        e=xn+e;
end
```

Nous pouvons nous apercevoir que dans la boucle for, nous calculons les coefficients a_n et b_n.

Nous pouvons synthétiser le signal créneau pour différent nombre de coefficients.

- Cas où n = 5 :

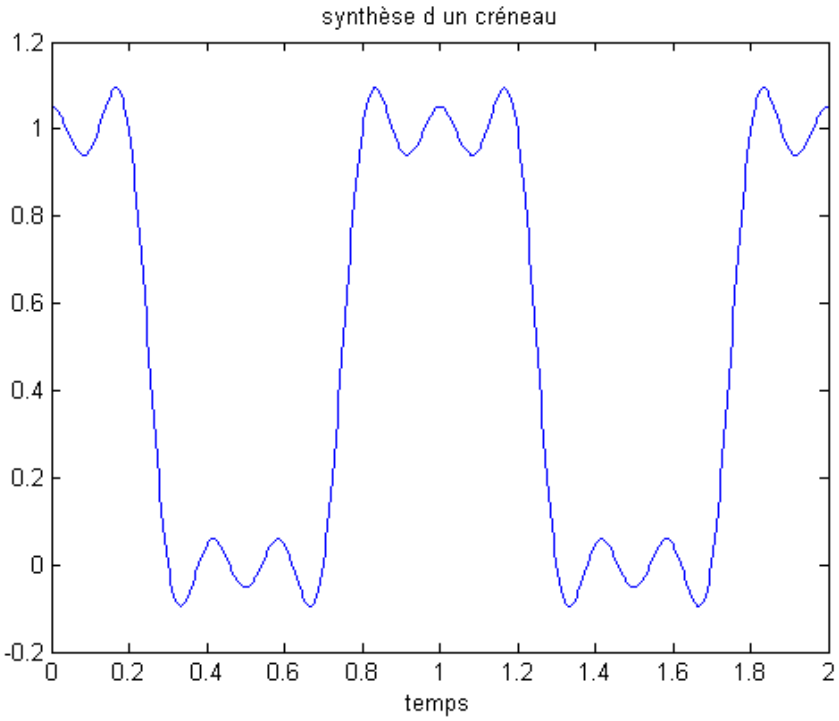


Figure 6 : Signal créneau pour 5 coefficients

- Cas où n = 15 :

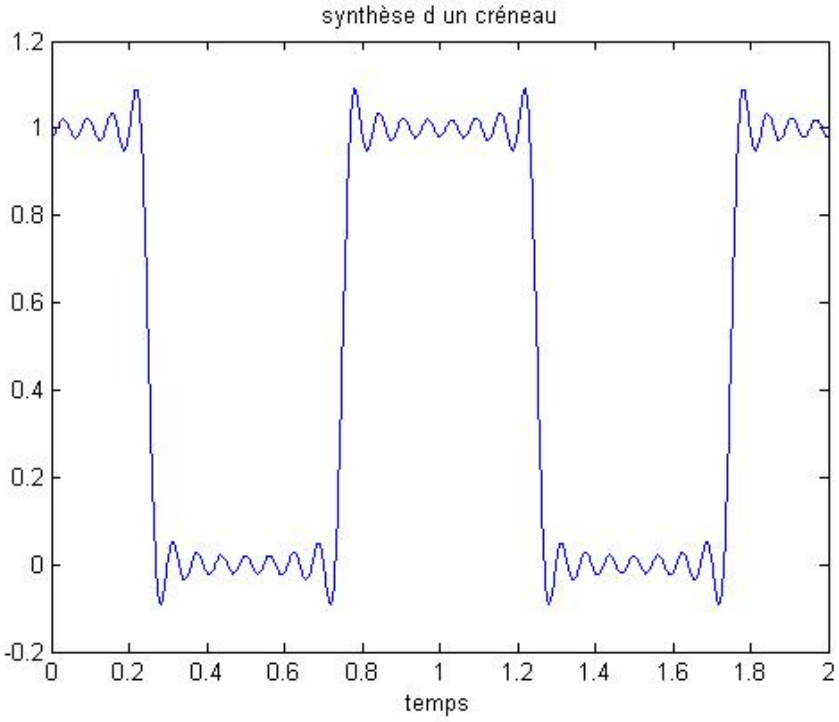


Figure 7 : Signal créneau pour 15 coefficients

- Cas où n = 50 :

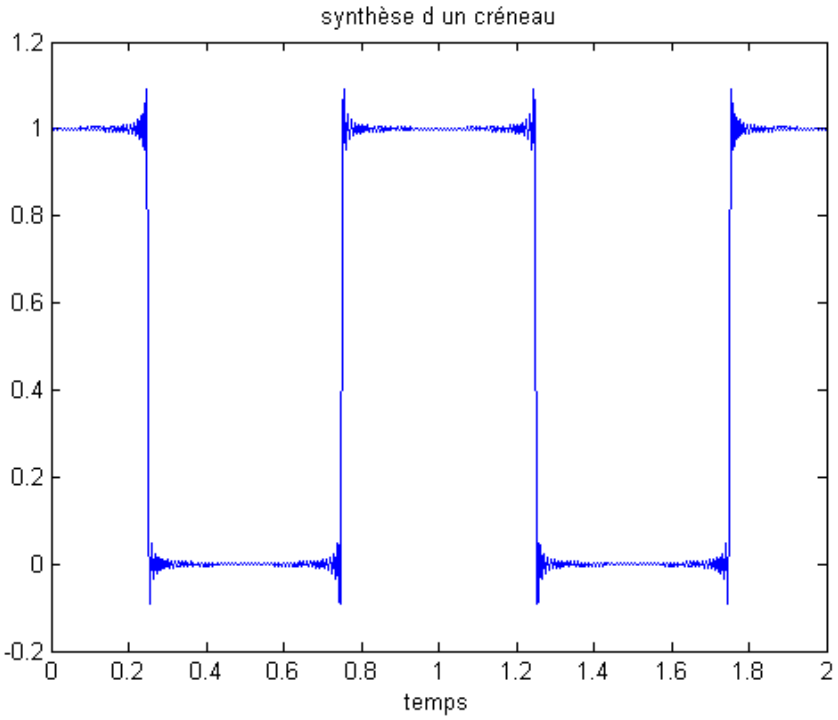


Figure 8 : Signal créneau pour 100 coefficients

2.4.2.2 *Cas d'un signal en dent de scie :*

Connaissant l'équation des coefficients C_n et sachant que le signal est impair donc les $a_n = 0$, alors nous pouvons en déduire l'expression de b_n :

$$b_n = -\frac{2.(-1)^n}{\pi.n}$$

Le programme permettant de calculer la décomposition en série de Fourier d'un signal dent de scie est le suivant :

```
n=input('Entrer le nombre de coefficient n pour la synthèse ');
dt=1./(50*n*f0);          % Incrément temporel
t=(0:dt:2/f0);             % Création du temps sur deux périodes du signal par incrément de dt
A0=A/2;                    % Valeur moyenne du signal
e=A0;                      % Boucle de calcul de la somme
for i=1:n
        an=0;
        bn=(-2*((-1)^i))/(pi*i);
        xn=an*cos(2*pi*f0*i*t)+bn*sin(2*pi*f0*i*t);
        e=xn+e;
end
```

Nous obtenons alors pour un nombre de coefficients égal à 100, nous obtenons :

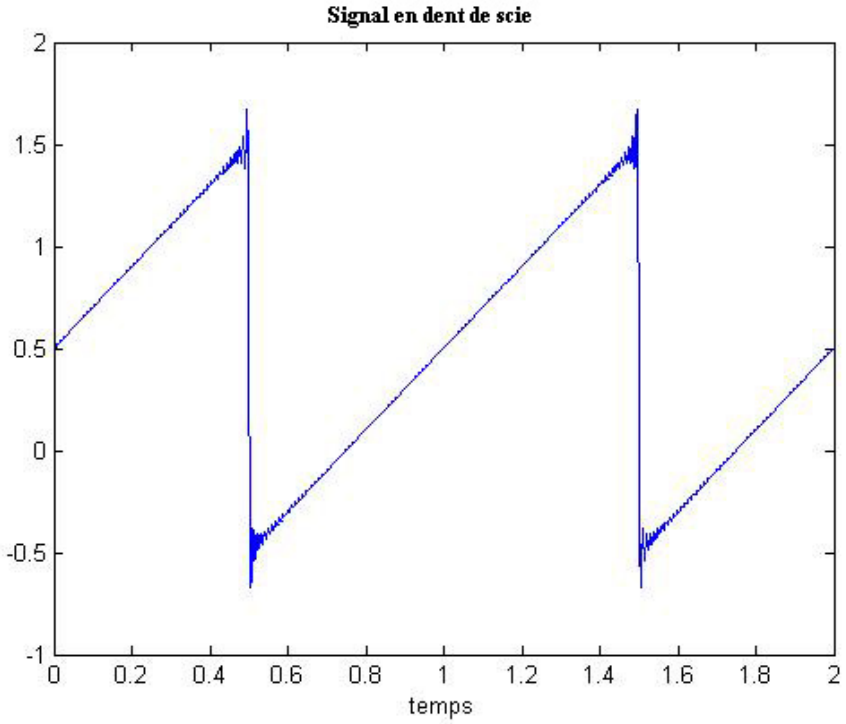


Figure 9 : Signal en dent de scie avec un nombre de coefficients égal à 100

2.5 Filtrage linéaire :

2.5.1 Analyse de l'influence de la fréquence de coupure :

2.5.1.1 Fréquence de coupure de 0.2 :

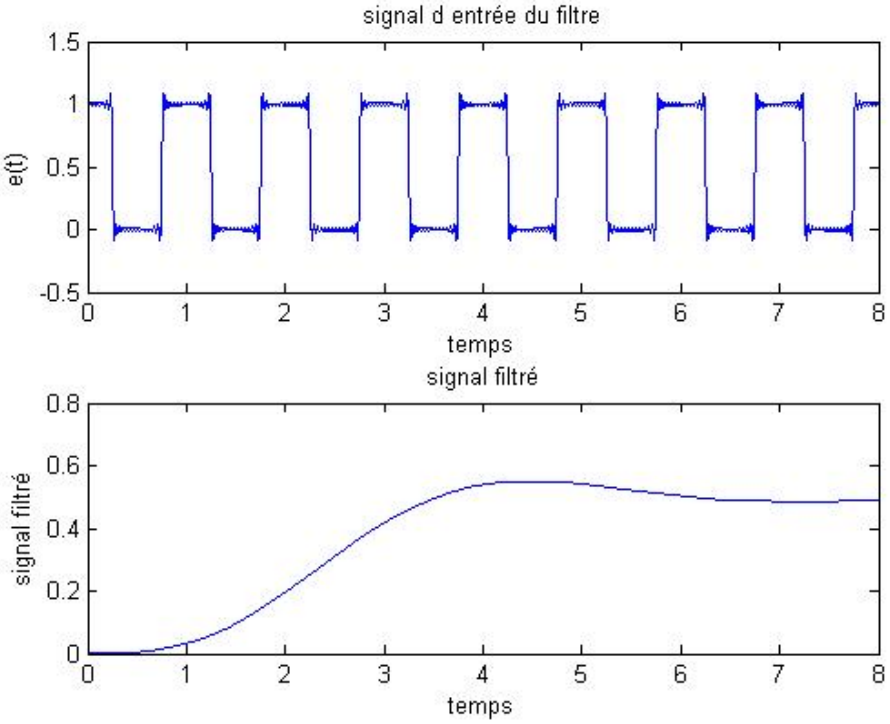


Figure 10 : Signal filtré avec une fréquence de coupure de 0,2

Nous pouvons remarquer que le signal filtré ne ressemble pas du tout au signal d'entrée.

2.5.1.2 Fréquence de coupure de 2 :

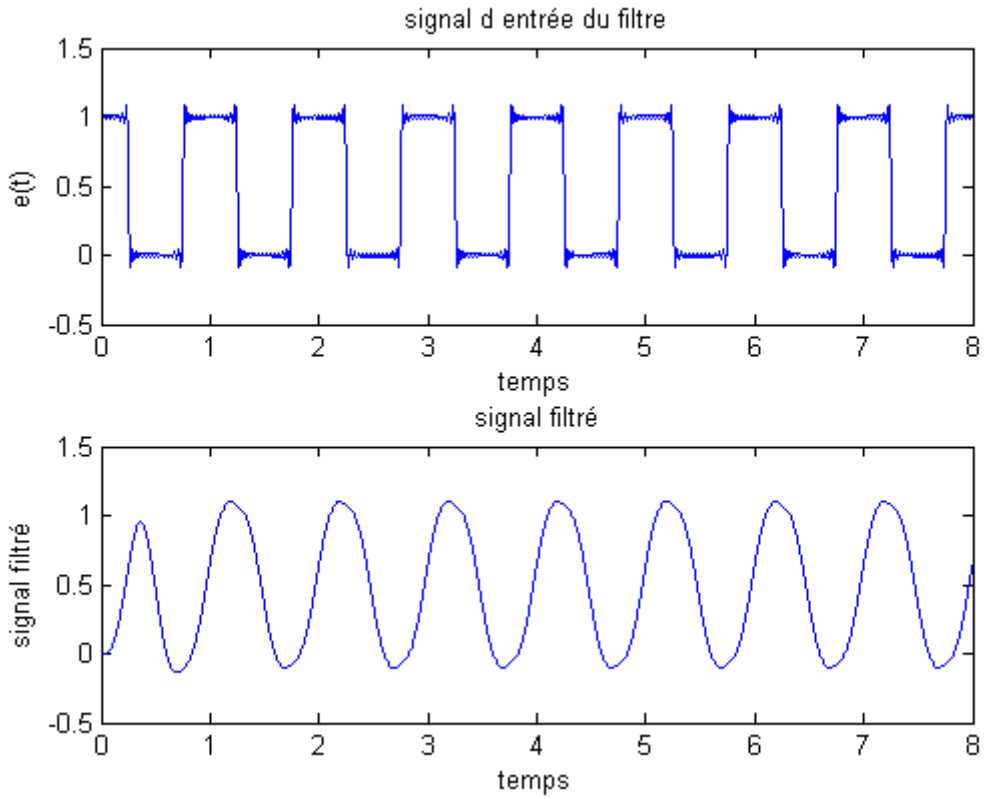


Figure 11 : Signal filtré avec une fréquence de coupure de 2

Nous pouvons constater que la fréquence de coupure n'est pas assez élevé pour apercevoir un signal correct en sortie du filtre.

2.5.1.3 Fréquence de coupure de 4 :

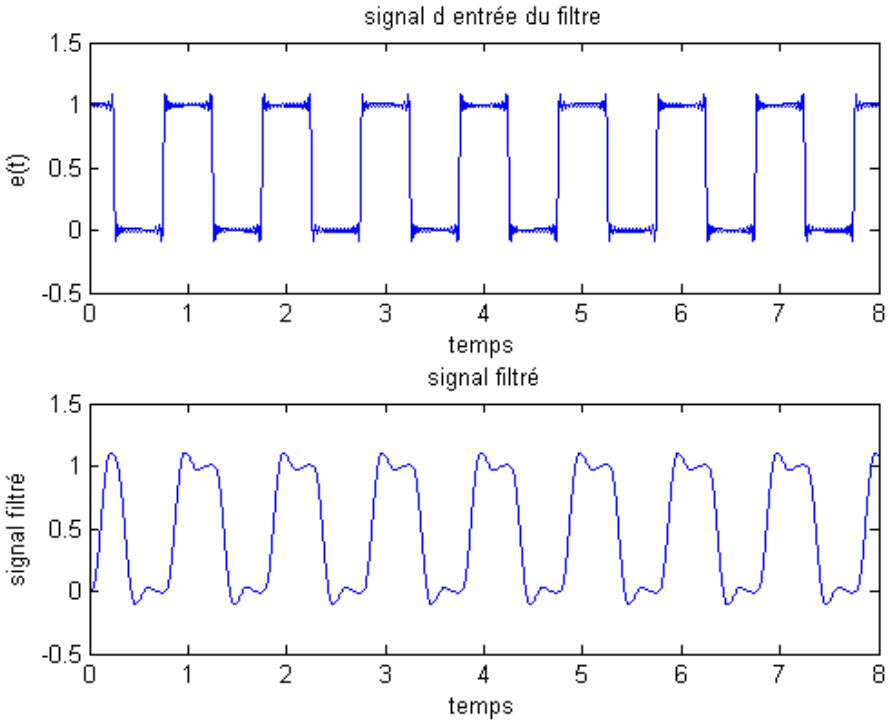


Figure 12 : Signal filtré avec une fréquence de coupure de 4

Nous pouvons remarquer que nous commençons d'apercevoir le signal réel en sortie mais que la fréquence de coupure est toujours trop faible.

2.5.1.4 Fréquence de coupure de 6 :

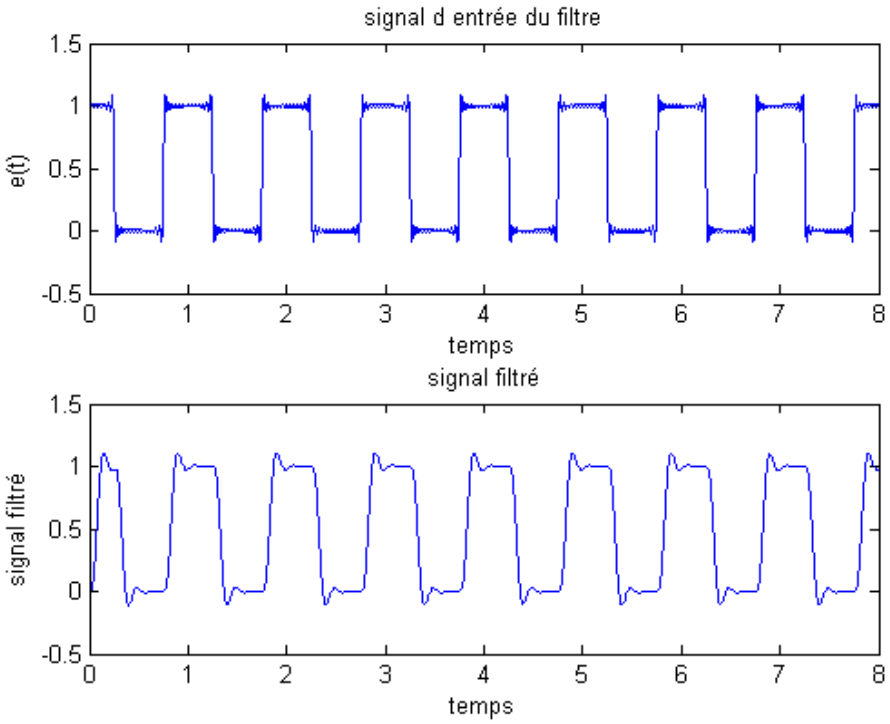


Figure 13 : Signal filtré avec une fréquence de coupure de 6

Nous pouvons constater que le signal est bien le signal d'entrée filtré. La fréquence de coupure est assez élevée

2.5.1.5 Fréquence de coupure de 8 :

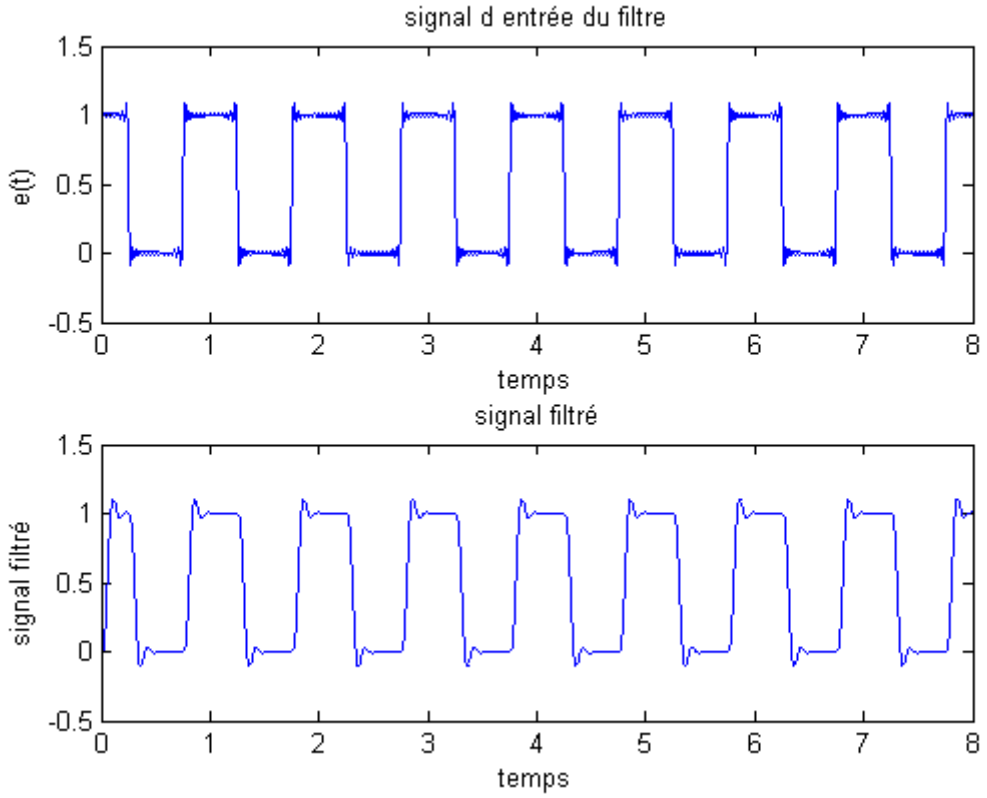
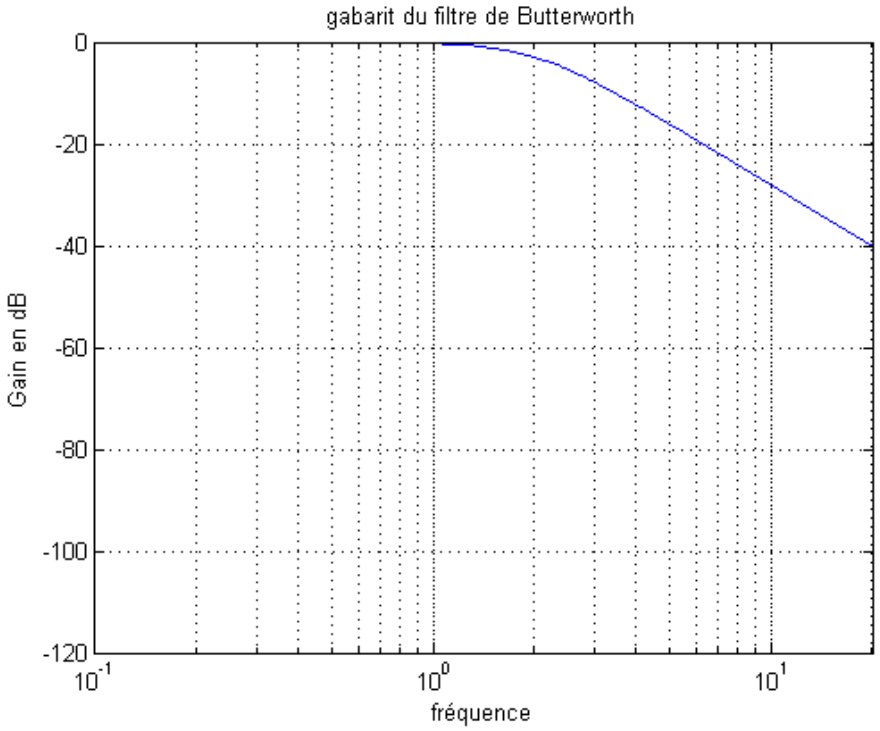


Figure 14 : Signal filtré avec une fréquence de coupure de 8

Nous pouvons constater que le signal est bien le signal d'entrée filtré. La fréquence de coupure est assez élevée

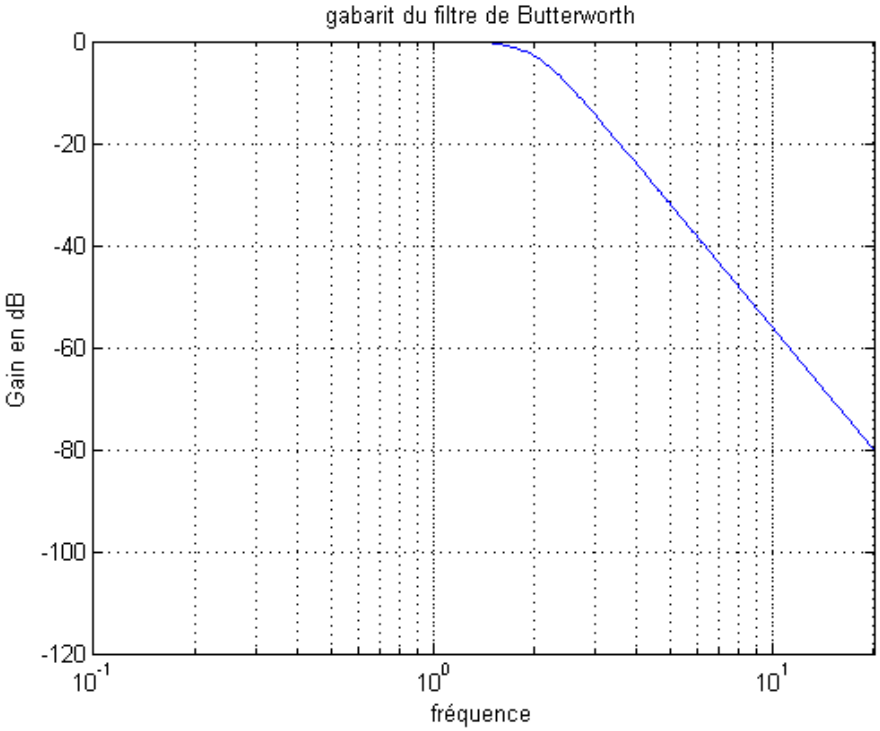
2.5.2 Analyse de l'influence de l'ordre du filtre :

2.5.2.1 Filtre d'ordre 2 :



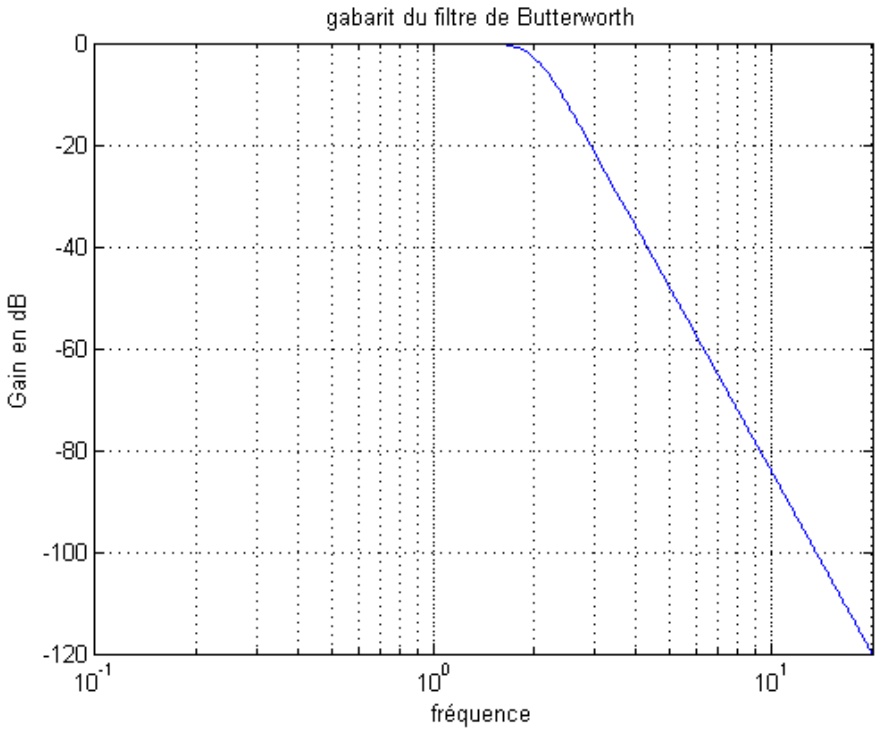
La pente est de–40 dB/dec.

2.5.2.2 Filtre d'ordre 4 :



La pente est de–80 dB/dec.

2.5.2.3 Filtre d'ordre 6 :



La pente est de–120 dB/dec.

2.5.3 Synthétisation d'un signal bruité puis filtré :

Nous avons synthétisé un signal créneau bruité puis filtrer pour éliminer le bruit :

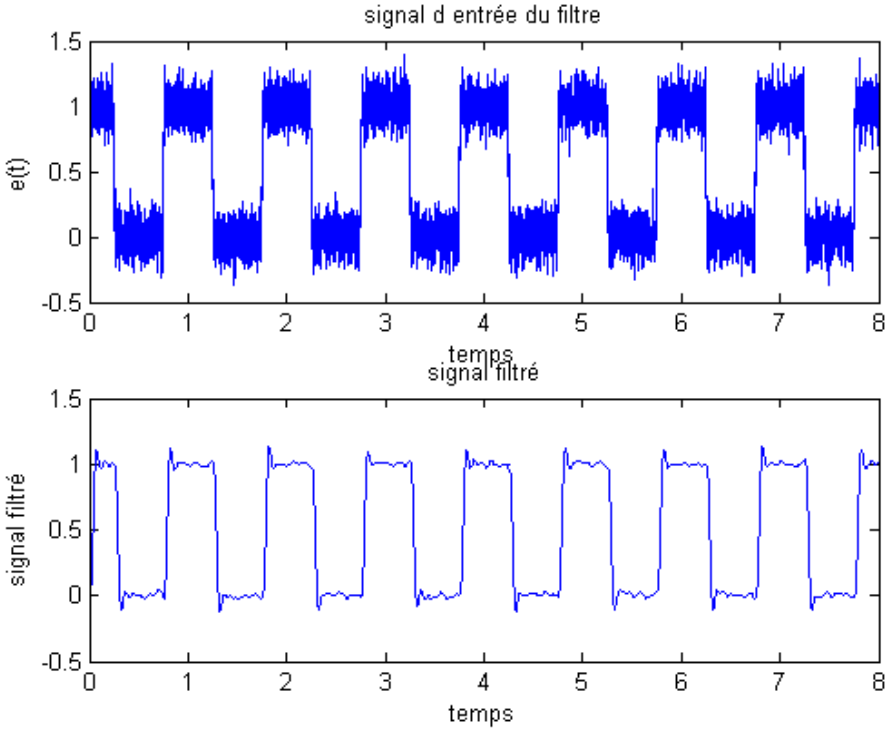


Figure 15 : Signal créneau bruité puis filtré

Le programme permettant un tel résultat est le suivant :

```
f0=1;                    % Fréquence du fondamental
A=1;                     % Amplitude du signal
n=25;                    % Nombre d'harmonique pour la synthèse du créneau
dt=1/(50*n*f0);          % Incrément temporel
t=(0:dt:8/f0);           % Création du temps sur 8 périodes du signal par incrément de dt
A0=A/2;                  % Valeur moyenne du signal
e=A0;
B=0.16;
bruit=B*randn(size(t));
for i=1:n
        an=A*sinc(i/2);
        xn=an*cos(2*pi*f0*i*t);
        e=xn+e;
end
e=e+bruit
figure(1)
subplot(2,1,1)
plot(t,e)
xlabel('temps')
ylabel('e(t)')
title('signal d entrée du filtre')
fn=1/(2*dt)                              % Moitié de la fréquence d'échantillonnage=Freq de Nyquist
fc=input('entrer la fréquence de coupure du filtre');      % Fréquence de coupure du filtre
ordre=input('entrer l ordre du filtre');   % Ordre du filtre
[CB,CA]=butter(ordre,fc/fn);             % Détermination des coef du filtre numérique
                                         % correspondant au Passe Bas d'ordre n de
                                         % fréquence de coupure fc
[H,f]=freqz(CB,CA,15000,1/dt);           % Obtention du gabarit du filtre H et fréquences associées f
figure(2)
semilogx(f,20*log10(abs(H)))             % Tracé du Gabarit du filtre
axis([0.1 10*fc  -120 0])
grid
title('gabarit du filtre de Butterworth')
xlabel('fréquence')
ylabel('Gain en dB')
ef=filter(CB,CA,e);                      % Filtrage du signal e avec le passe bas
figure(1)
subplot(2,1,2)
plot(t,ef)                               % Tracé du signal filtré
title('signal filtré')
xlabel('temps')
ylabel('signal filtré')
```