Compte Rendu de TP 01

Traitement Numérique du Signal

Nom : BOUARICHE

Prénom : Iheb

Année : 2023/2024

Spécialité : Instrumentation an2

Séance 01 :

I. Signal sinusoïdal et échantillonnage, DSP, signal carré :

a. Visualisation de données successives :

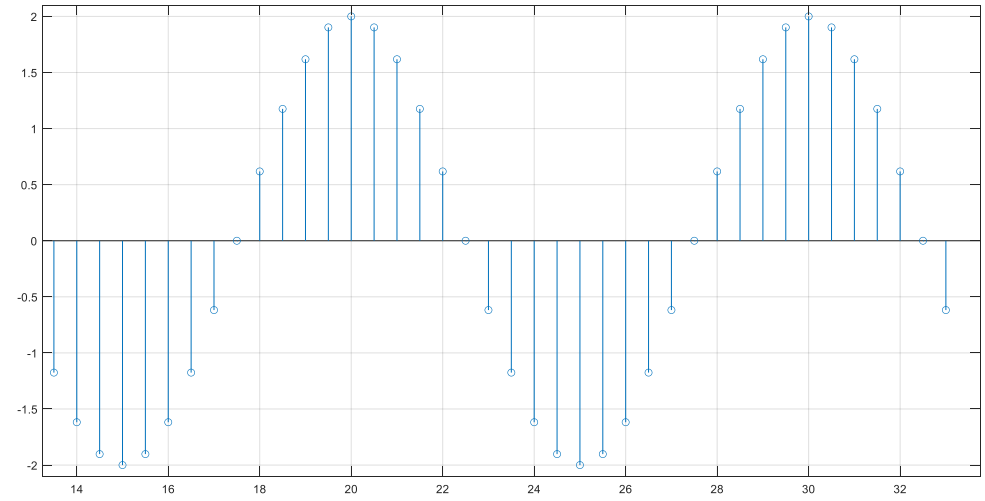


Figure : Signal sinusoïde échantillonné en fonction du temps.

- La période d'échantillonnage Te = 0.5

- La fréquence d'échantillonnage Fe = 1/Te = 2

- La période de la sinusoïde T = 10

- La fréquence de la sinusoïde T = 1/10 = 0.1

On peut voir que ce signal sinusoïde est de période T (et fréquence F) et aussi un signal échantillonné avec une période Te (fréquence Fe).

b. Visualisation de données au cours du temps :

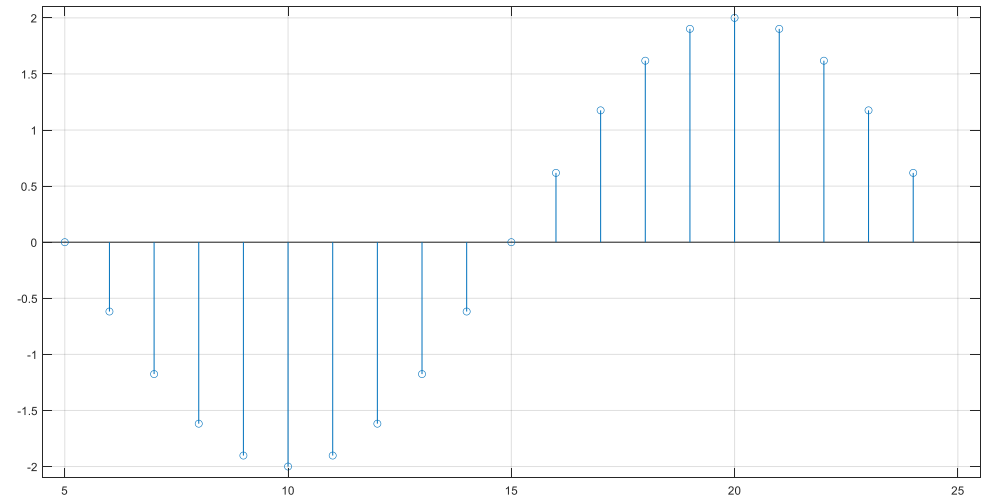


Figure : Signal sinusoïde échantillonné en fonction du temps.

- La période d'échantillonnage Te = 1

- La fréquence d'échantillonnage Fe = 1/Te = 1

- La période de la sinusoïde T = 20

- La fréquence de la sinusoïde T = 1/20 = 0.05

Le changement de l'instruction va changer la période du signal sinusoïde (T devient 20) et ainsi la fréquence (devient 0.05).

Le rôle de cette instruction justifie ça : $t = TP1\_2(F, Fe, durée, periode)$; Tel que Fe est la fréquence d'échantillonnage et F la période de signal sinusoïde.

c. Échantillonnage d'un signal sinusoïdal, DSP :

On considère un signal sinusoïdal causal à temps continu:

$$x(t) = a\sin(\omega_0 t + \phi)$$

Avec : $\omega_0 = 2\pi F_0;\ a = 2\ et\ \phi = \pi/2$

I.3.

```
% Pour fe = 75Hz

    a=2;
    f0=50;
    fe=75;
    tn=0:1/fe:(100e-3-1/fe);
    xn=a*sin(2*pi*f0*tn+pi/2);
    figure(1);
    stem(tn,xn);
    xlabel('Temps(s)')
    ylabel('Amplitude(u.a)')
    title('Signal échantillonné par une fréquence de 75Hz')
    grid on
```

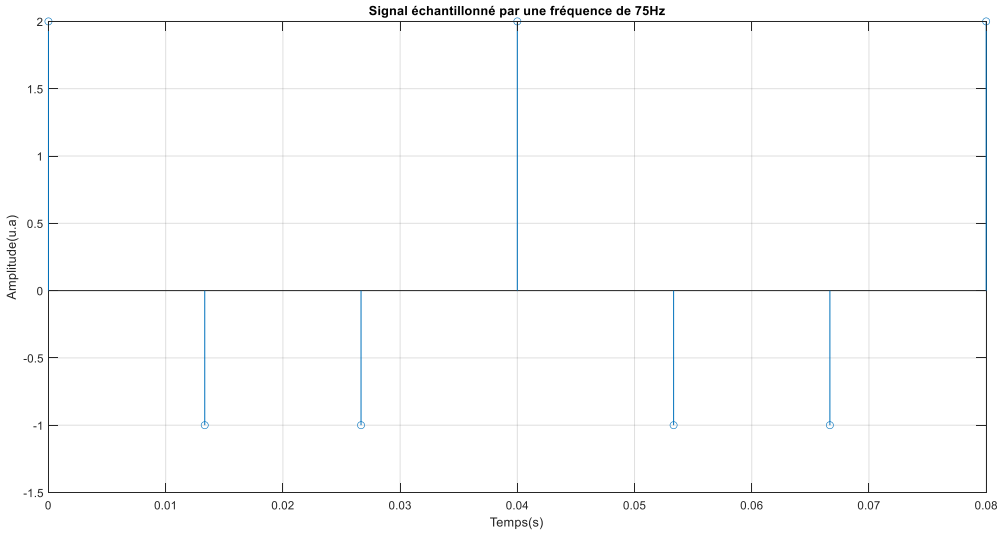


Figure : Signal sinusoïde échantillonné en fonction du temps.

Commentaire : On peut voir qu'on a perdu beaucoup d'informations avec une fréquence d'échantillonnage qui est très faible. Et on ne peut pas avoir des informations sur le signal originale. Le théorème de Shannon n'est pas vérifié et on peut pas revenir au signal original à partir de ce signal.

La fréquence d'échantillonnage ($Fe = 75Hz$) < 2 fois la fréquence de signal original ($F0 = 50Hz$)

```
% pour fe = 150Hz

      a=2;
      f0=50;
      fe=150;
      tn=0:1/fe:(0.1-1/fe);
      xn=a*sin(2*pi*f0*tn+pi/2);
      figure(2);
      stem(tn,xn);
      xlabel('Temps(s)')
      ylabel('Amplitude(u.a)')
      title('Signal échantillonné par une fréquence de 150Hz')
      grid on
```

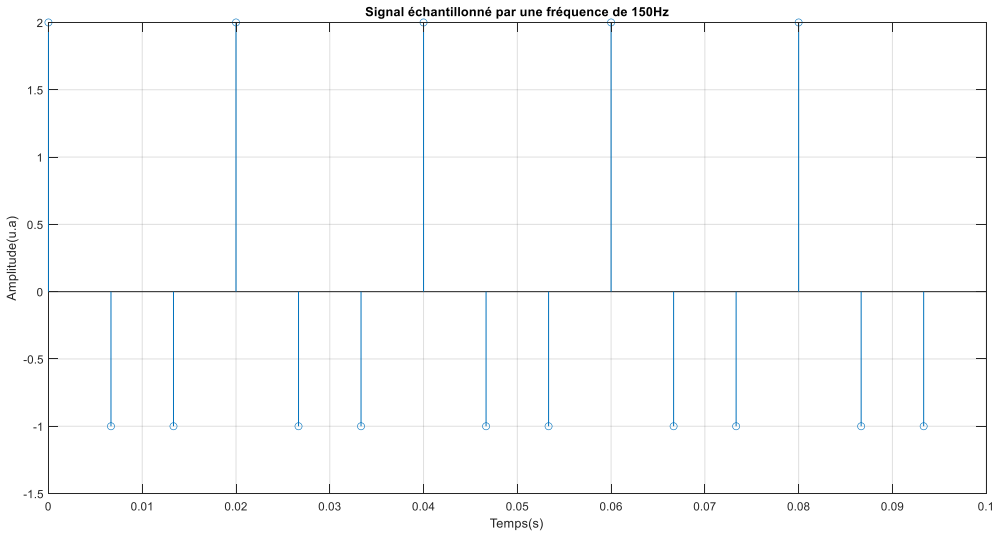


Figure : Signal sinusoïde échantillonné en fonction du temps.

Commentaire : Visuellement on ne peut pas voire clairement les informations du signal original car dans ce cas on a aussi la fréquence d'échantillonnage qui est faible. Mais le théorème de Shannon est vérifié et on peut revenir au signal original à partir de ce signal.

La fréquence d'échantillonnage ($Fe = 150Hz$) > 2 fois la fréquence de signal original ($F0 = 50Hz$)

```
% pour Fe = 1KHz

      a=2;
      f0=50;
      fe=1e3;
      tn=0:1/fe:(100e-3-1/fe);
      xn=a*sin(2*pi*f0*tn+pi/2);
      figure(3);
```

```
stem(tn,xn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title('Signal échantillonné par une fréquence de 1KHz')
grid on
```

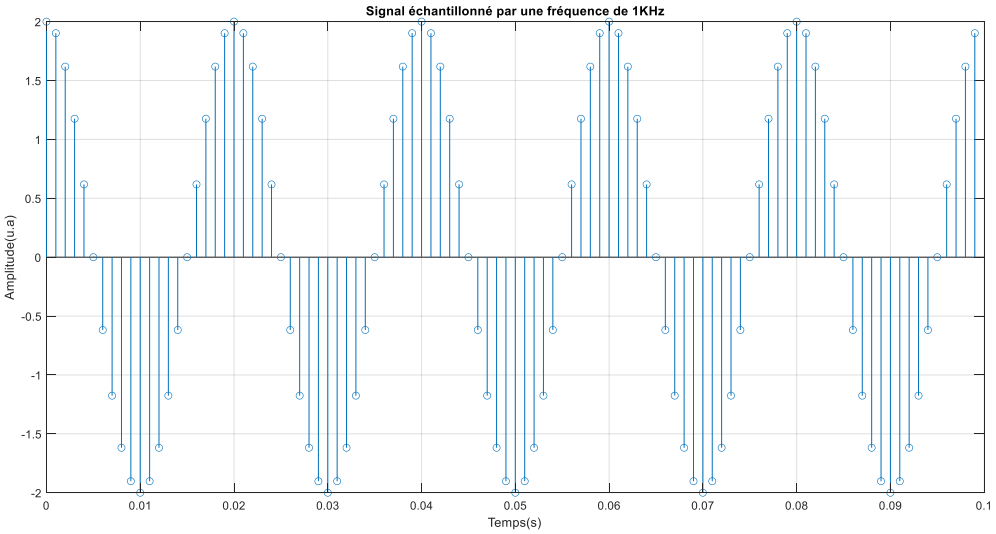


Figure : Signal sinusoïde échantillonné en fonction du temps.

Commentaire : Visuellement on peut voire clairement les informations du signal original car dans ce cas la fréquence d'échantillonnage est très élevée. Et le théorème de Shannon est vérifié et c'est sûr qu'on peut revenir au signal original à partir de ce signal.

La fréquence d'échantillonnage ($Fe = 1KHz$) >> 2 fois la fréquence de signal original ($F0 = 50Hz$)

I.4

Le script :

```
a=2;
f0=50;
fe=500;
Nq = 2^8
tn=0:1/fe:(40e-3-1/fe);
xn=a*sin(2*pi*f0*tn+pi/2);
figure(4);
stem(tn,xn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title("Signal échantillonné et quantifié d'une fréquence de 500Hz et
sur 8 bits")
fprintf("la valeur du pas de quantification = %f \n", a*2/Nq)
grid on

b=3;
Q=2*a/(2^b);
xq=(floor(xn/Q)+0.5)*Q;
figure(5);
plot(tn,xn,tn,xq);
xlabel('Temps(s)')
```

```
ylabel('Amplitude(u.a)')
title("Signal échantillonné vs signal quantifié")
grid on

err=mean(abs(xn-xq));
fprintf("l'erreur de quantification = %f \n", err)
fprintf("l'erreur de quantification varie en fonction de variantion
de buffer. Si on augmente le nombre de buffer on aura plus de
précision \n")

a=2; f0=50; fe=500; tn=0:1/fe:(40e-3-1/fe);
xn=a*sin(2*pi*f0*tn+pi/2);
b_l=1:12;
Q_l=2*a./(2.^b_l);
err_l=zeros(size(b_l));
for Q_=1:length(Q_l)
  Q=Q_l(Q_);
  xq=(floor(xn/Q)+0.5)*Q;
  err_l(Q_)=mean(abs(xn-xq));
end
figure(6); semilogy(b_l,err_l);
xlabel('Temps(s)')
ylabel('Erreur')
title("L'erreur moyenne de quantification")
grid on
```

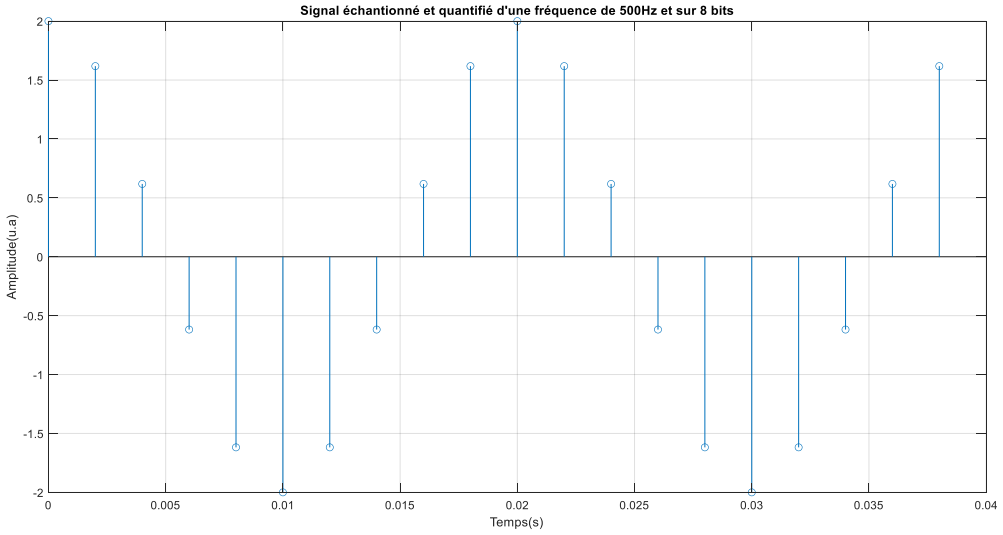


Figure : Signal sinusoïde échantillonné en fonction du temps.

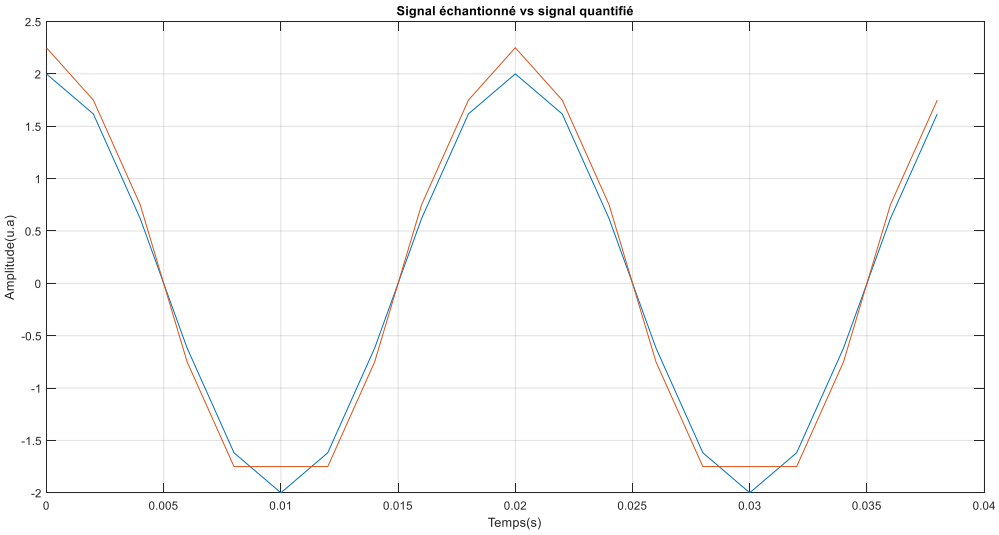


Figure : Signal échantillonné vs signal quantifié.

- ✓ La valeur du pas de quantification = 0.015625
- ✓ L'erreur de quantification = 0.155573
- ✓ L'erreur de quantification varie en fonction de variation de buffer. On constate que Si on augmente le nombre de buffer on aura plus de précision

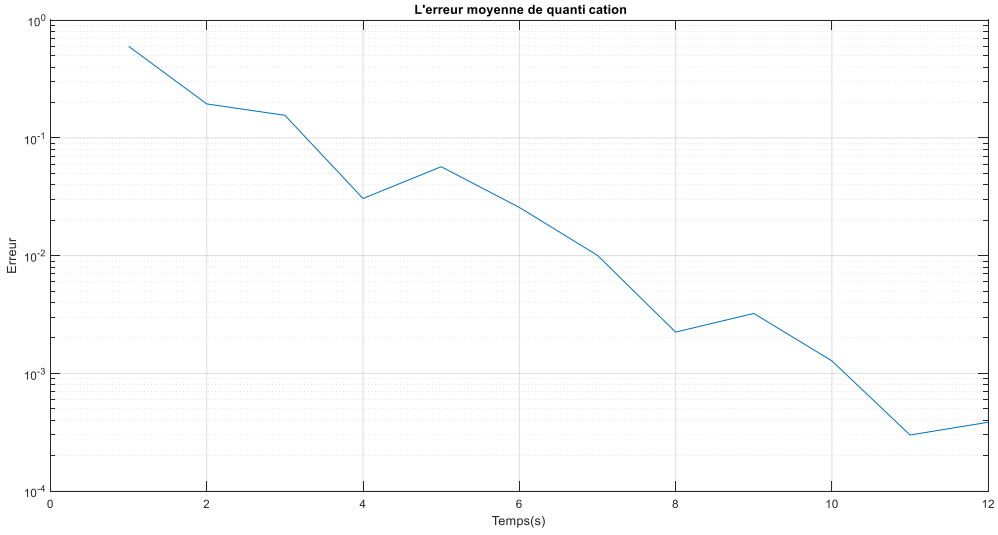


Figure : L'erreur moyenne quadratique.

1.5. La DSP d'un signal discret x_n :

$$S(f_k) = \frac{1}{N}|X(f_k)|^2$$

Où :

- $S(f_k)$, est la densité spectrale de puissance au point de fréquence discrète f_k.
- $X(f_k)$, est la transformée de Fourier discrète du signal x_n.
- N, est la taille de la séquence x_n.

1.6

```
a=2;
f0=50;
fe=100.2;
tn=0:1/fe:(3-1/fe);
xn=a*sin(2*pi*f0*tn+pi/2);
figure(8);
stem(tn,xn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title('Signal échantillonné par une fréquence de 100.2Hz')
grid on
```

✓ On remarque qu'on a eu une partie de signal et ça c'est parce que on a une petite fréquence d'échantillonnage et ça peut être vérifie par le théorème de Shannon

1.7

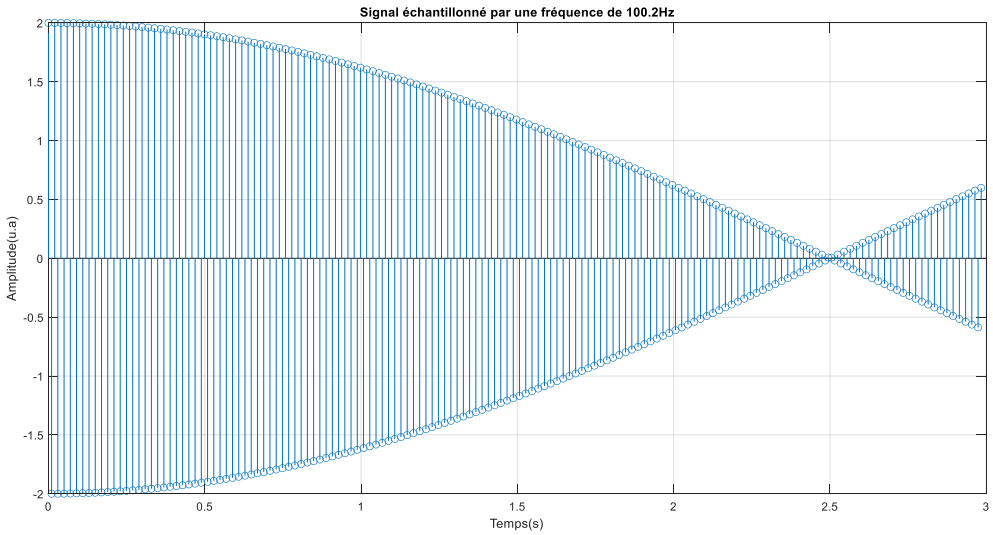


Figure : Signal échantillonné par une fréquence de 100.2 Hz.

```
feShannon = f0*2;
```

✓ On ne peut pas bien voire le signal sinusoïde graphiquement. Mais la valeur de la fréquence d'échantillonnage doit être supérieur à la valeur de la fréquence de Shannon : Fe = 100, et on peut vérifier que ce signal contient tous les informations fréquentiel de signal original et on peut revenir au signal original.

1.7 La condition de Shannon-Nyquist est vérifié car Fe = 100.2 > 2*F0 = 2*50 = 100.

1.8

```
Xf= fft(xn);
n = length(xn);
f = (0:n-1)*(fe/n);
power = abs(Xf).^2/n;
figure(9)
plot(f,abs(power));
grid on
```

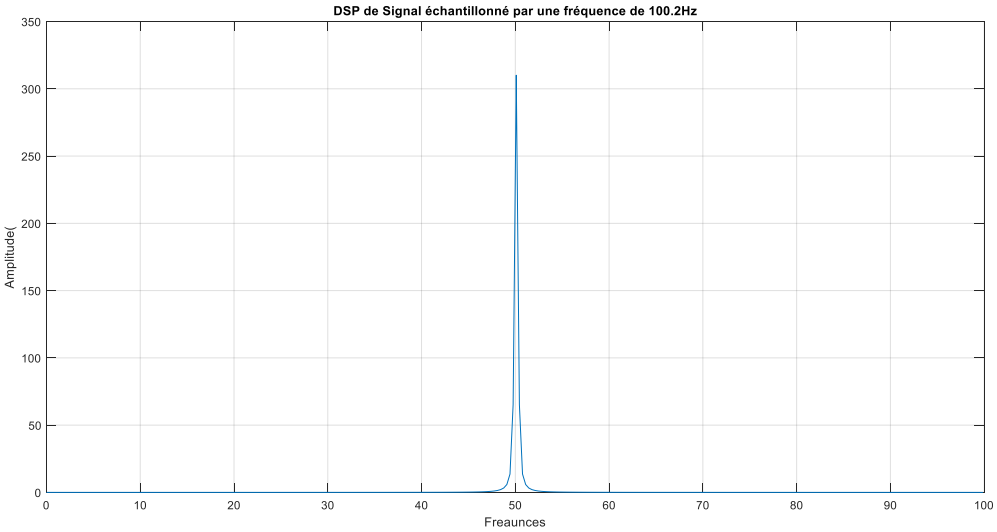


Figure : DSP de signal échantillonné par une fréquence de 100.2Hz.

On peut bien voire que le pic est autour de la fréquence de 50 Hz ce qui égale la fréquence de signal. Dans le cas d'un signal avec une fréquence d'échantillonnage très grande (presque continue) la largeur de pic va être presque nul et on peut avoir seulement une impulsion de Dirac. Donc la largeur de cette impulsion c'est à cause de l'échantillonnage.

d. Visualisation de données au cours du temps lorsque la fréquence d'échantillonnage est faible par rapport à la fréquence de la sinusoïde :

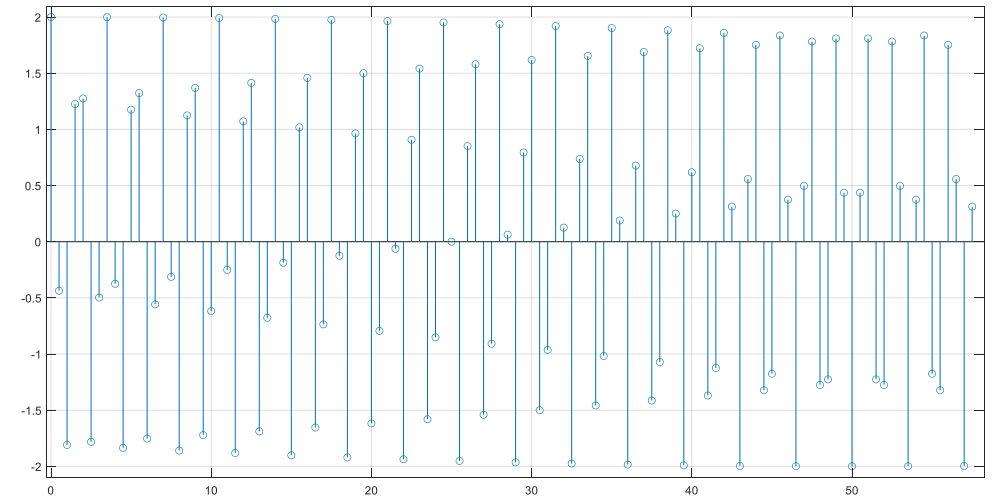


Figure : Signal échantillonnée en fonction du temps.

Commentaire : Ce signal perd beaucoup d'informations comparant au signal original, car la fréquence d'échantillonnage est beaucoup plus faible par rapport à la fréquence de signal.

e. Signal carré et DSP :

```
f0=50;
fe = 500;
tn=0:1/fe:(40e-3-1/fe);
```

```
xn=square(pi*2*f0*tn);
figure(10)
stem(tn,xn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title('Signal échantionné par une fréquence de 500Hz')
grid on


fs = 500;                % sampling frequency
t = 0:(1/fs):(10-1/fs); % time vector
S = xn;
n = length(S);
X = fft(S);
f = (0:n-1)*(fs/n);     %frequency range
figure(11);%power
fshift = (-n/2:n/2-1)*(fs/n); % zero-centered frequency range
powershift = abs(X).^2/n;
powershift = fftshift(powershift);% zero-centered power
plot(fshift,powershift)
grid on
```

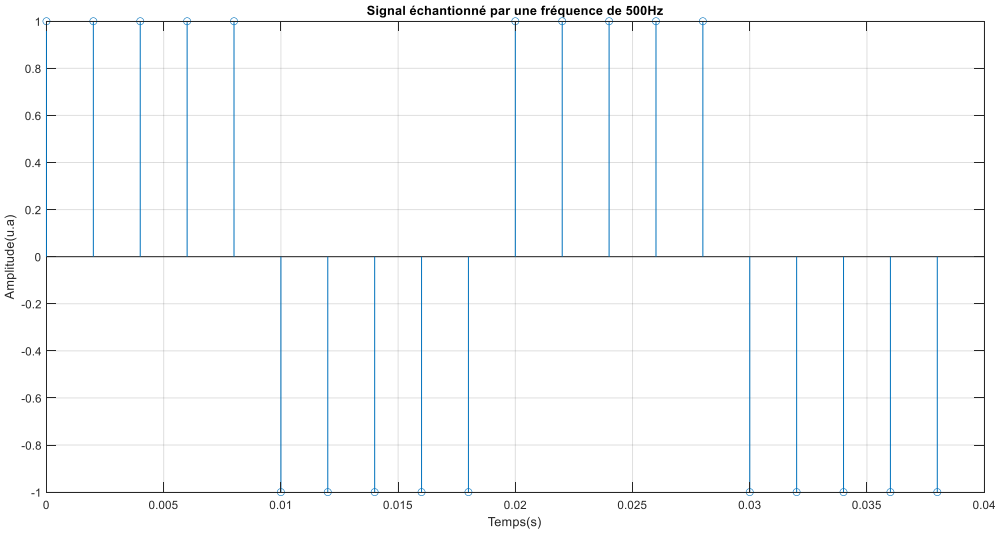
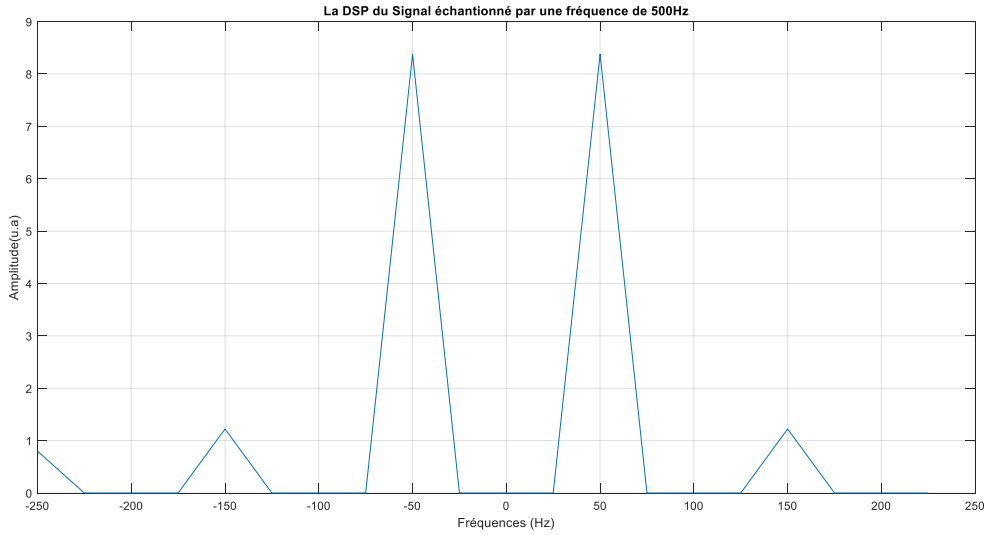


Figure : Signal échantillonnée en fonction du temps.



I.11 La fréquence de la valeur de la DSP corresponds à la fréquence nulle est 0 car la moyenne de notre signal (Signal carré) est nul.

I.12 La fréquence du second terme de la DSP = La fréquence d'échantillonnage / 10 = 50.

I.13 La fréquence du dernier terme de la DSP = La fréquence d'échantillonnage /3.33 = 150.

I.14 La différence entre un DSP d'signal sinusoïdal et d'un signal carré :

- Le signal sinusoïdal peut être représenté avec un pic pour une seule fréquence car on a une seule sinusoïde de fourrier. Et on aura une certaine largeur à cause de l'échantillonnage et ces pics vont être périodique avec une fréquence d'échantillonnage Fe.

- Le signal Carré peut être réalisé avec une superposition de plusieurs sinusoïdes de fourrier avec des fréquences différentes, Mais la fréquence de la périodicité de signal carré a la valeur de l'amplitude maximale. Et puisque ce signal est échantillonné on a un signal de DSP périodique par une fréquence Fe.

f. Signal et bruit blanc :

```
yn = square(0.3)*randn(1,length(tn))+ xn;
figure(12)
stem(tn,yn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title('Signal échantionné par une fréquence de 500Hz')
grid on

fs = 500;                % sampling frequency
t = 0:(1/fs):(10-1/fs); % time vector
S = yn;
n = length(S);
X = fft(S);
f = (0:n-1)*(fs/n);     %frequency range
figure(13);%power
fshift = (-n/2:n/2-1)*(fs/n); % zero-centered frequency range
powershift = abs(X).^2/n;
powershift = fftshift(powershift);% zero-centered power
plot(fshift,powershift)
grid on
```

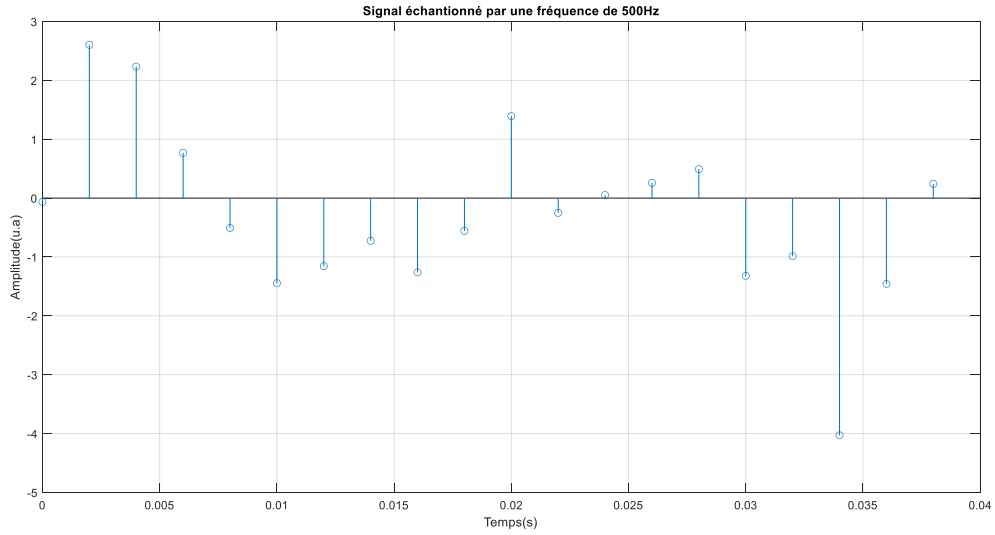


Figure : Signal échantillonnée.

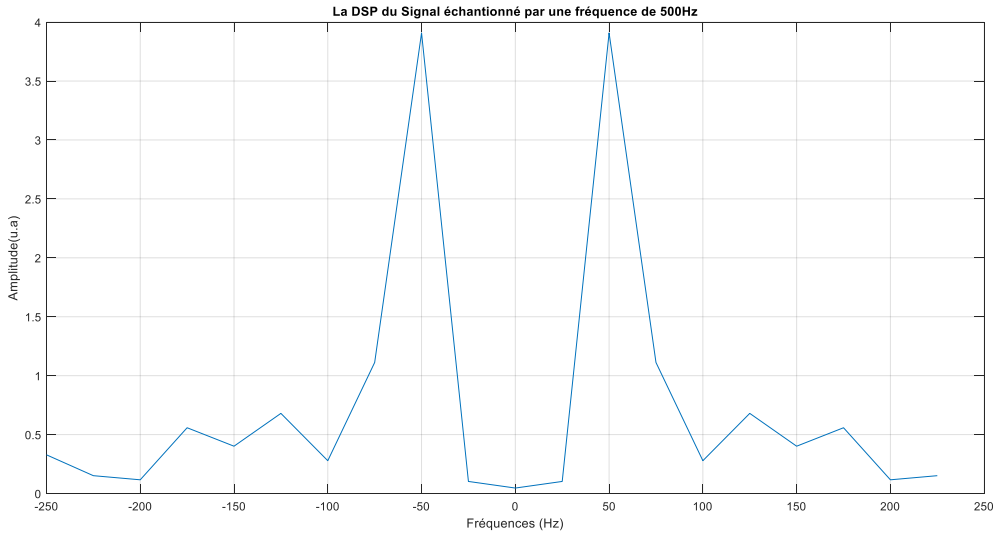


Figure : La DSP du signal.

I.15 Explication des résultats : On voie ici la différence par rapport au signal sinusoïde sans bruit et avec bruit. Dans ce cas à la présence de bruit on a eu des amplitudes pour les fréquences qui sont supérieur à la fréquence de signal, cela explique les fréquences de bruit qui sont grandes et leurs effets sur le DSP de signal original.

I.16 On peut utiliser ici un filtre passe bas pour diminuer l'effet de bruit sur le signal.

<u>Séance 02 :</u>

II. Fonction de corrélation, détection radar :

1.

```
% Premiere partie
%%%%%%%%%% I.e %%%%%%%%%%
a=2;
f0=50;
```

```
fe=500;
tn=0:1/fe:(100e-3-1/fe);
xn=a*sin(2*pi*f0*tn+pi/2);
figure()
stem(tn,xn);
xlabel('Temps(s)')
ylabel('Amplitude(u.a)')
title('Signal échantionné par une fréquence de 500Hz')
grid on

fs = 500;                % sampling frequency
t = 0:(1/fs):(10-1/fs); % time vector
S = xn;
n = length(S);
X = fft(S);
figure();
fshift = (-n/2:n/2-1)*(fs/n); % zero-centered frequency range
powershift = (abs(X).^2)/n;
powershift = fftshift(powershift);% zero-centered power
xlabel('Frequences')
ylabel('Amplitude')
title('spectre du signal échantionné par une fréquence de 500Hz')
plot(fshift,powershift)
grid on
```

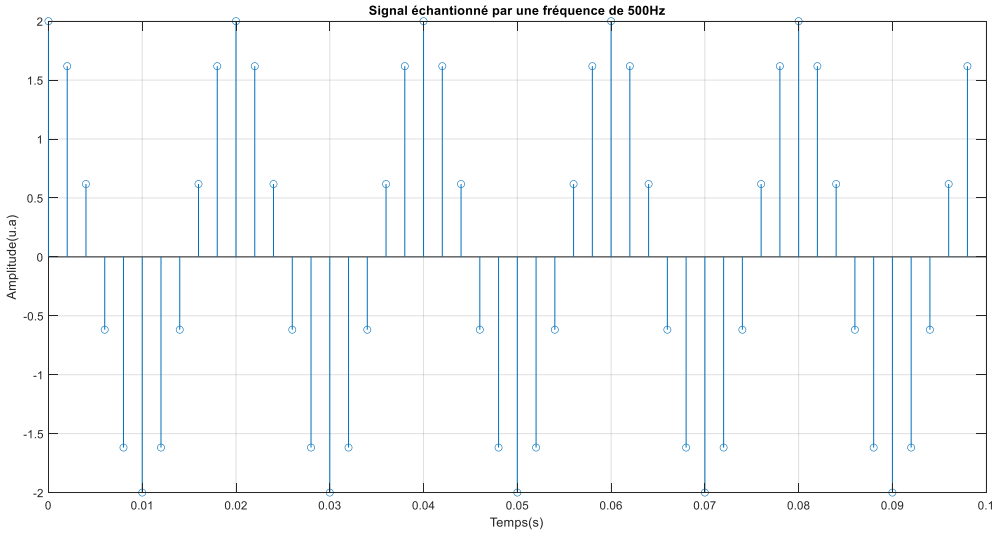


Figure : Signal échantillonnée en fonction du temps.

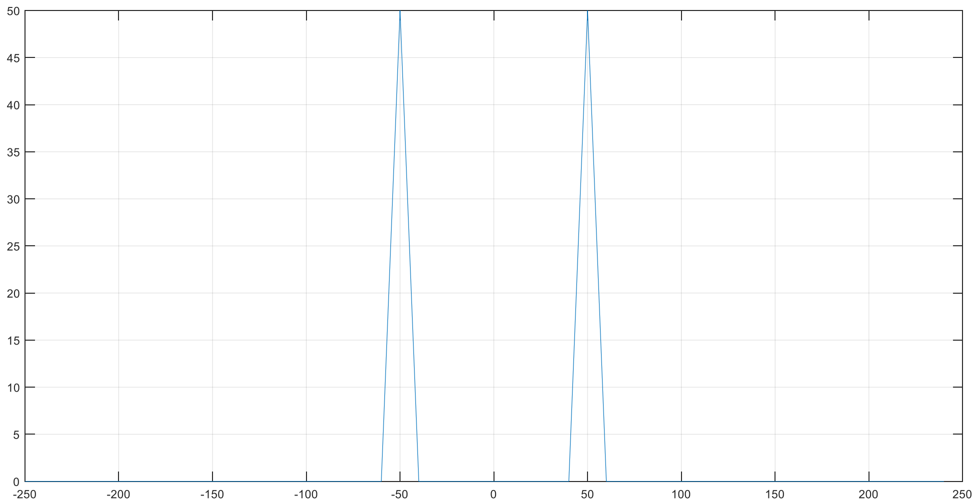


Figure : la DSP du signal.

On constate deux pics a la fréquence de signal et une largeur à cause de l'échantillonnage.

2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% le cas de 0.5
bn1 = square(0.5)*randn(1,length(tn));
yn1 = bn1 + xn;
% le cas de 2
bn2 = square(2)*randn(1,length(tn));
yn2 = bn2 + xn;
% le cas de 5
bn3 = square(5)*randn(1,length(tn));
yn3 = bn3 + xn;
```

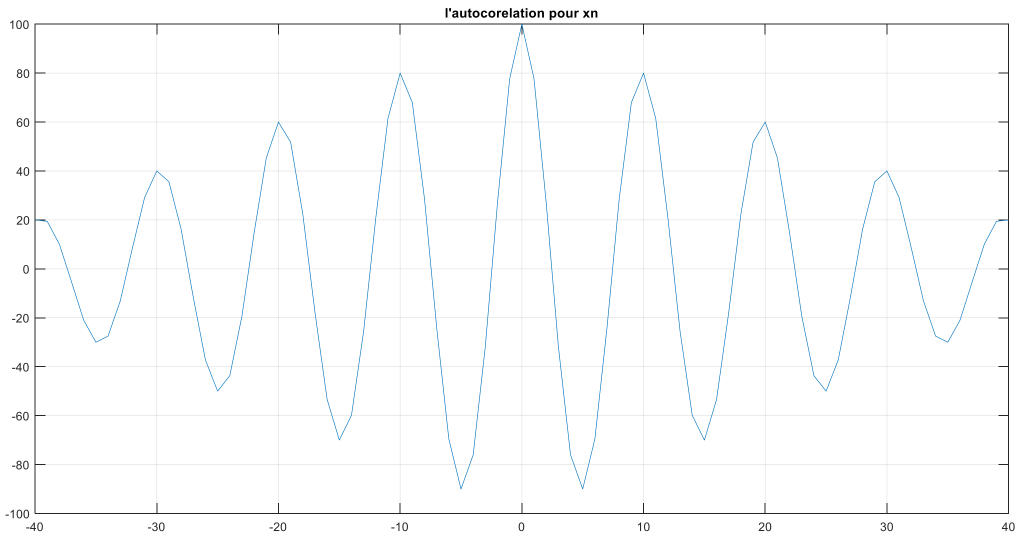


Figure : signal d'autocorrélation d'un signal sinusoïde.

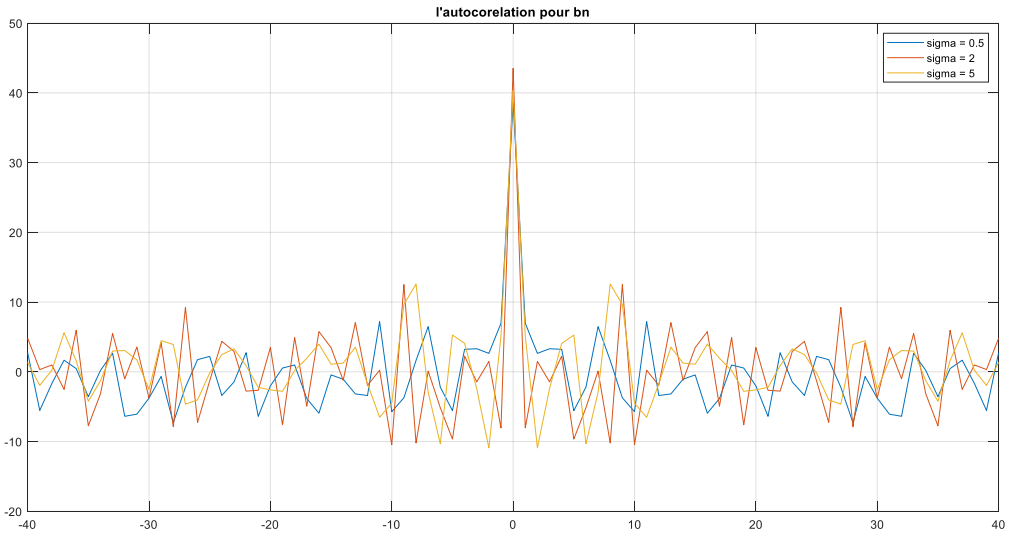


Figure : signal d'autocorrélation d'un signal bruit.

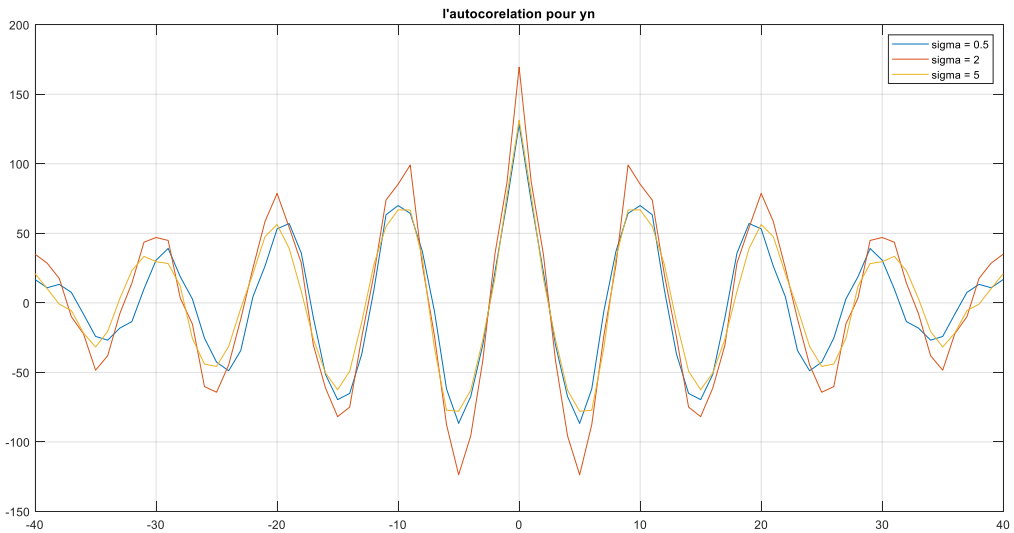


Figure : signal d'autocorrélation de yn.

On peut voir que l'autocorrélation du bruit est mieux détectable par rapport les autres. Et on peut voir qu'à zéro on aura un pic d'une amplitude maximale. On peut aussi remarquer que le bruit n'a pas vraiment un grand effet sur la détection et sur la forme de signal d'autocorrélation.

3.

```
% II.3.
m1 = randn(1,50);
m2 = sin(2*pi*(1:50)*0.1);
m3 = sin(2*pi*(1:50)*0.1).*exp(-(-24:25).^2/10^2);


[Cm1,LAGSm1] = xcorr(m1,m1,L);
[Cm2,LAGSm2] = xcorr(m2,m2,L);
[Cm3,LAGSm3] = xcorr(m3,m3,L);
```

```
figure()
plot(l,Cm1)
grid on
title("l'autocorelation pour la fonction randn")
figure()
plot(l,Cm2)
grid on
title("l'autocorelation pour la fonction sinus")
figure()
plot(l,Cm3)
grid on
title("l'autocorelation pour la fonction sinusoïde modulée par une
gaussienne")
```

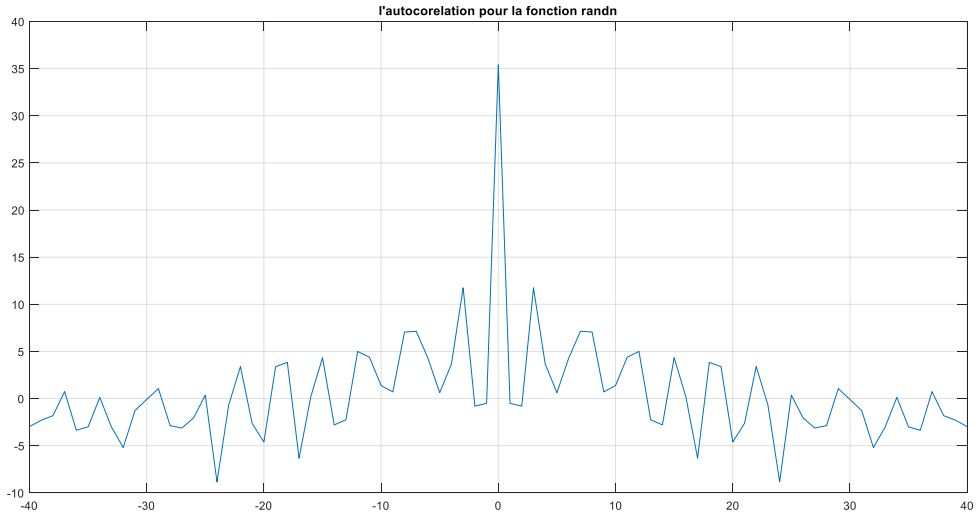


Figure : Signal d'autocorrélation bruit.

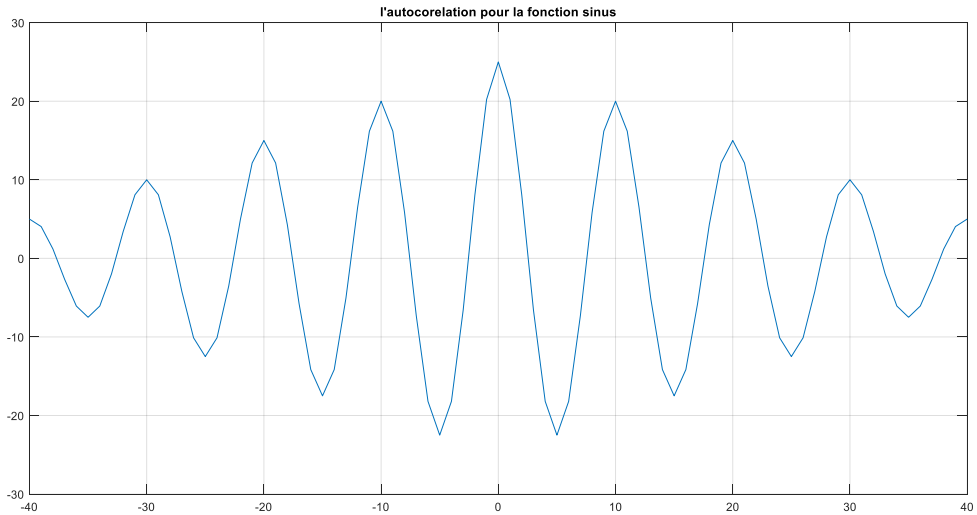


Figure : Signal d'autocorrélation de signal sinus.

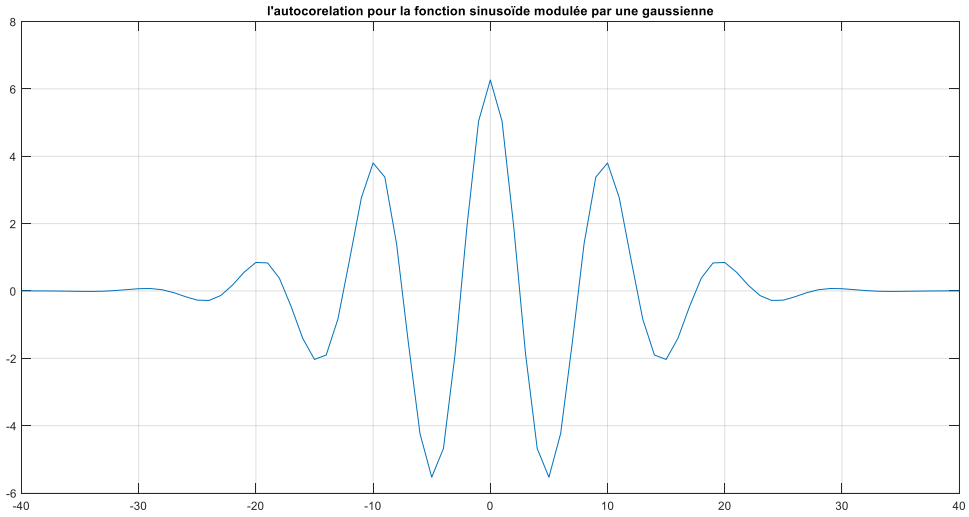


Figure : Signal d'autocorrélation sinusoïde modulé par une gaussienne.

On peut voir que l'autocorrélation de bruit avec lui-même est interprétable et on peut faire la détection d'un signal mieux que le signal sinusoïdal ou le signal sinusoïdal bruité, car ou peut voire clairement pour l'autocorrélation de bruit qu'au point zéro on a les deux signaux sont identiques et aussi car on a une valeur maximale.

4.

```
% II.4
m4 = [m1,zeros(1,50),m1,zeros(1,100),m1];
figure()
plot(0:1:length(m4)-1,m4);
grid on
title("Le signal envoyé par le radar sans bruit")


m5 = m4 + 0.1*randn(1,length(m4));
figure()
plot(0:1:length(m5)-1,m5);
grid on
title("Le signal reçu par le radar bruité ")
```

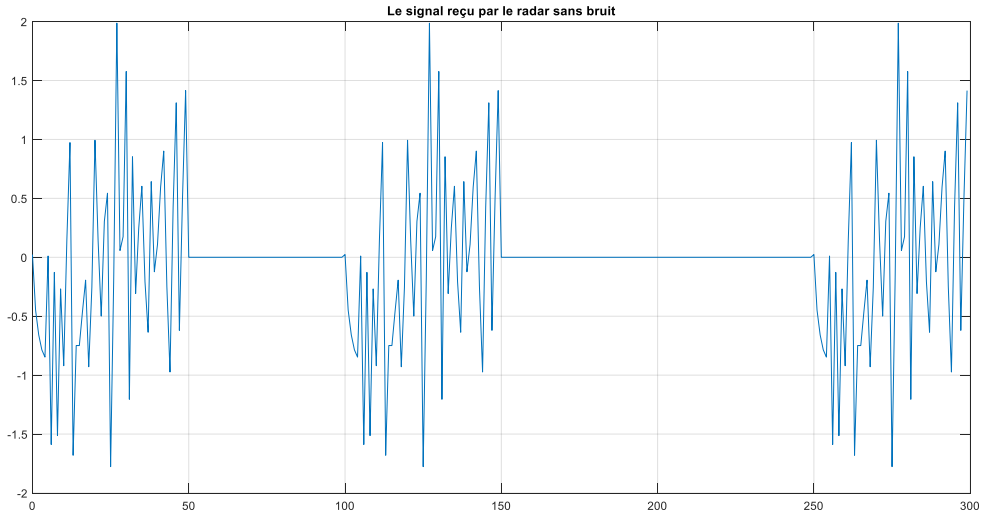


Figure : Le signal de radar sans bruit.

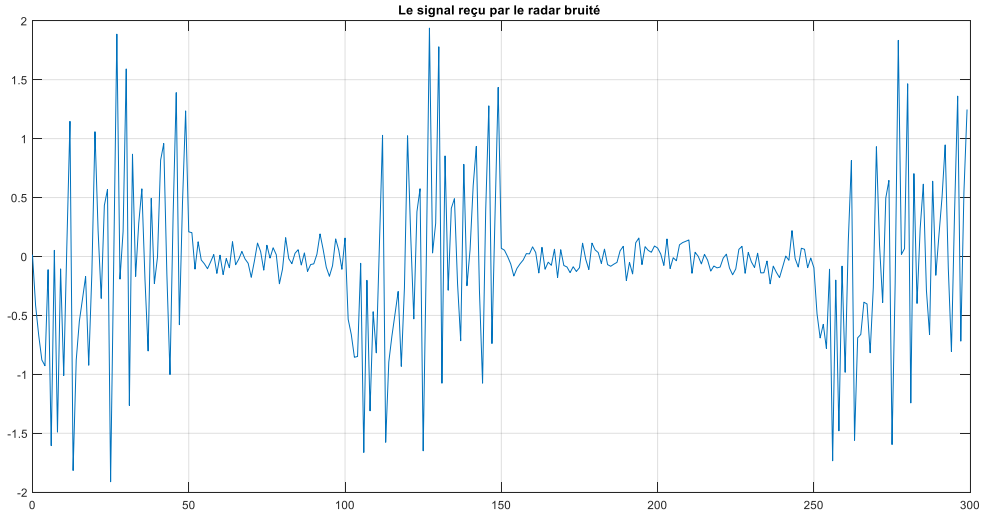


Figure : Le signal reçu par le radar avec bruit.

5.

```
% le scan du signal
m4 = [m1,zeros(1,50),m1,zeros(1,100),m1];
m5 = m4 + 0.1*randn(1,length(m4));
[Cm5,] = xcorr(m5,m1);
figure()
% plot(l,Cm1)
grid on
hold on
plot(-299:1:299,Cm5)
% legend("Autocorellation du bruit blanc","Autocorellation du bruit
blanc de longueur 300")
title("l'autocorelation entre le signal motif et le signal bruité")
```

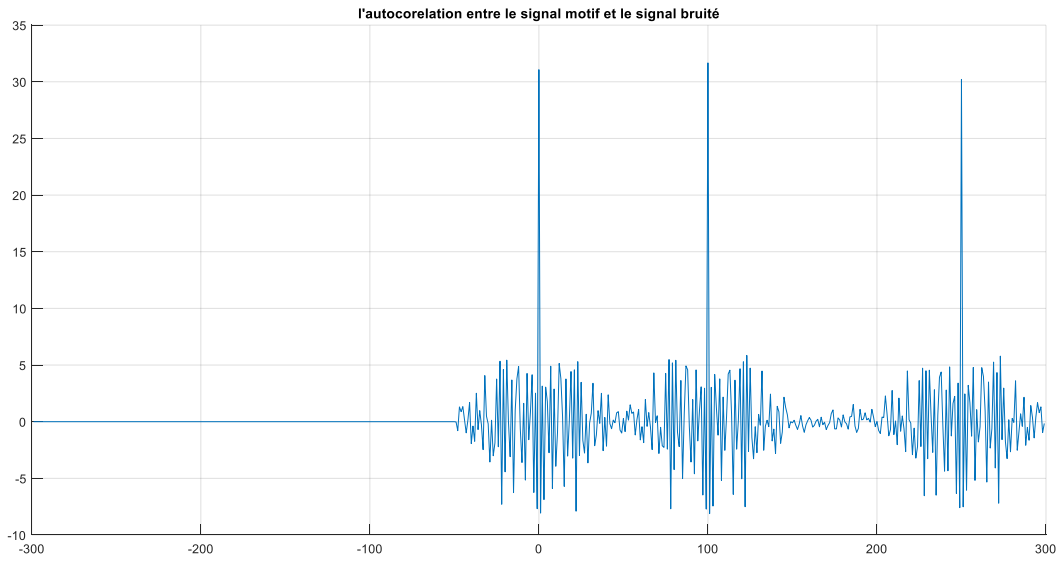


Figure : L'autocorrélation entre le signal motif et le signal bruité.

Observation : On remarque qu'on a bien détecter la position des trois cibles et avec ses positions [0, 100, 250], et ça vérifie le principe d'intercorrélation pour la détection. Lorsque le signal de bruit soit superposé sur la même position de bruit dans le signal, le résultat de l'intercorrélation entre ces deux représentent résulte une valeur maximale.

6.

```
% le scan du signal
m4 = [m1,zeros(1,50),m1,zeros(1,100),m1];
m5 = m4 + 1*randn(1,length(m4));
[Cm5,] = xcorr(m5,m1);
figure()
% plot(l,Cm1)
grid on
hold on
plot(-299:1:299,Cm5)
% legend("Autocorellation du bruit blanc","Autocorellation du bruit
blanc de longueur 300")
title("l'autocorelation entre le signal motif et le signal bruité")
```

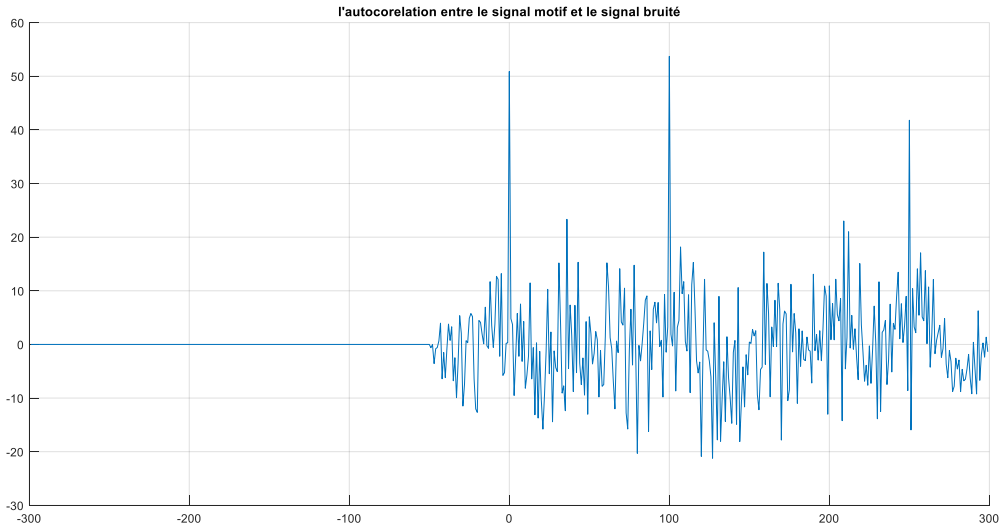


Figure : l'autocorrélation entre le signal motif et le signal bruité.

```
% le scan du signal
m4 = [m1,zeros(1,50),m1,zeros(1,100),m1];
m5 = m4 + 2*randn(1,length(m4));
[Cm5,] = xcorr(m5,m1);
figure()
% plot(l,Cm1)
grid on
hold on
plot(-299:1:299,Cm5)
% legend("Autocorellation du bruit blanc","Autocorellation du bruit
blanc de longueur 300")
title("l'autocorelation entre le signal motif et le signal bruité")
```

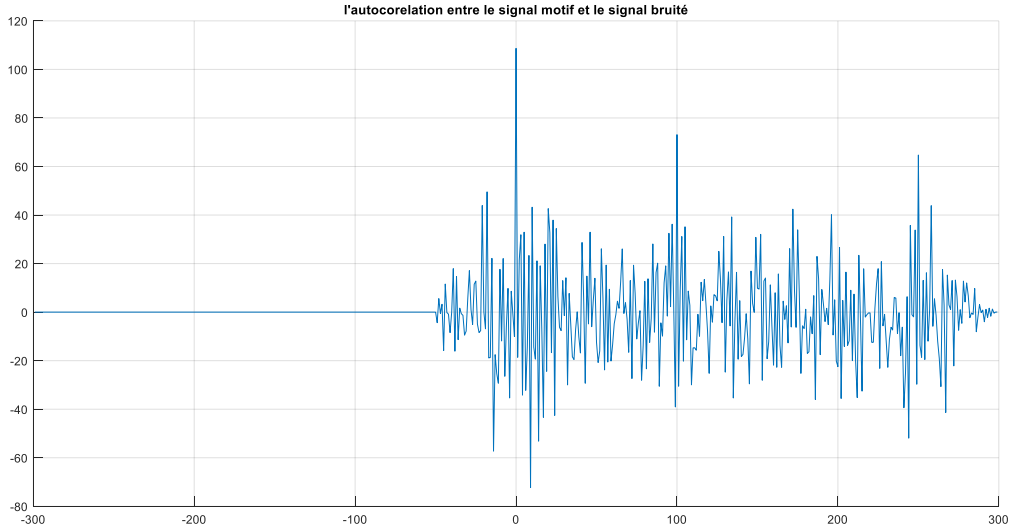


Figure : l'autocorrélation entre le signal motif et le signal bruité.

On constate que l'augmentation de niveau de bruit peut affecter le signal de sortie de l'intercorrélation et dégrade les résultats obtenus et même les performances de la détection.

Mais on peut voir que cette méthode est robuste même contre les perturbations qui sont moyennes. Et c'est ça le cas pour la plupart des applications.

On peut fixer une valeur « M » et on compare cette valeur avec toutes les valeurs, et si la valeur de signal d'intercorrélation est supérieure à « M », on détecte cette valeur avec sa position dans le temps.

7.

Pour un signal discret non-périodique et nul en dehors des instants $n = 0, \ldots, n = N-1$, la densité spectrale d'énergie $S_y(\omega)$ et la fonction d'autocorrélation $R_y(\omega)$ sont liées par la transformée de Fourier discrète (DFT) comme suit :

$$S_y(\omega) = \frac{1}{2\pi} \sum_{m=0}^{N-1} R_y(m) e^{-jwm}$$

8.

Ici, $S_y(\omega)$ est la transformée de Fourier discrète de la fonction d'autocorrélation R_y du signal $y[n]$, où n varie de 0 à $N-1$. La densité spectrale d'énergie décrit la répartition de l'énergie du signal dans le domaine fréquentiel, tandis que la fonction d'autocorrélation quantifie la corrélation entre les valeurs du signal à différents instants.

9.

Pour un signal discret périodique $x[n]$ de période N, la densité spectrale de puissance $P_x(\omega)$ et la fonction d'autocorrélation $R_y[m]$ sont liées par la transformée de Fourier discrète (DFT) comme suit :

$$P_x(\omega) = \frac{1}{N} \left| \sum_{m=0}^{N-1} R_x[m] e^{-jwm} \right|^2$$

10.

Ici, $P_x(\omega)$ est la transformée de Fourier discrète de la fonction d'autocorrélation $R_x[m]$ du signal $x[n]$ de période N. La densité spectrale de puissance décrit la distribution de la puissance du signal dans le domaine fréquentiel, tandis que la fonction d'autocorrélation quantifie la corrélation entre les valeurs du signal à différents instants.

11.

La fonction « xcorr » de MATLAB est utilisée pour calculer la corrélation croisée entre deux signaux, ce qui est utile pour mesurer à quel point les signaux sont similaires. Cependant, cette fonction ne peut pas être directement utilisée pour vérifier numériquement l'équation donnée précédemment, car la définition de la fonction « xcorr » dans MATLAB diffère de la formule de la densité spectrale de puissance en question.

La fonction « xcorr » calcule la corrélation croisée entre deux signaux en utilisant la somme des produits de leurs échantillons retardés, tandis que l'équation pour la densité spectrale de puissance implique une transformation de Fourier discrète de la fonction d'autocorrélation. Ainsi, bien que la

fonction « xcorr » puisse être utilisée pour comparer les similitudes entre deux signaux périodiques, elle ne fournit pas une représentation directe de la densité spectrale de puissance telle qu'indiquée dans l'équation.

Séance 03 :

III. Sur-échantillonnage d'un signal discret

a)

```
load("signalbase.mat")
fe = 25
Te = 1/fe;
T = 1/25:Te:length(x)/25;
figure()
% plot(l,Cm1)
grid on
hold on
plot(T,x)

% La DSP de Sn
fe = 25;                % sampling frequency
t = 0:(1/fe):(10-1/fe); % time vector
S = x;
n = length(x);
X = fft(x);
f = (0:n-1)*(fe/n);     %frequency range
figure();
fshift = (-n/2:n/2-1)*(fe/n); % zero-centered frequency range
powershift = abs(X).^2/n;
powershift = fftshift(powershift);
plot(fshift,powershift);
grid on

figure();
semilogx(fshift, powershift);
title('DSP de sn en fréquences centrées en Hertz');
xlabel('Fréquences (Hz)');
ylabel('DSP');
grid on;
```

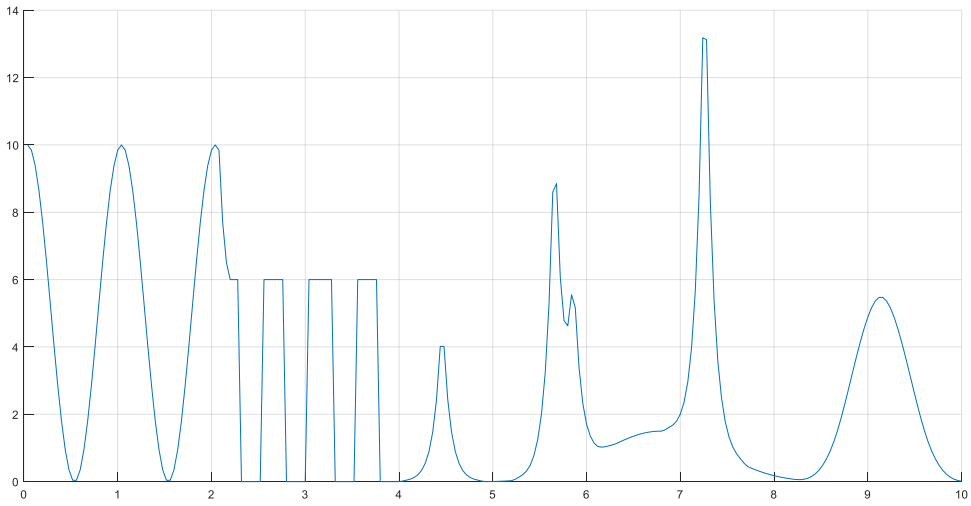


Figure : Le signal s(t) en fonction du temps.

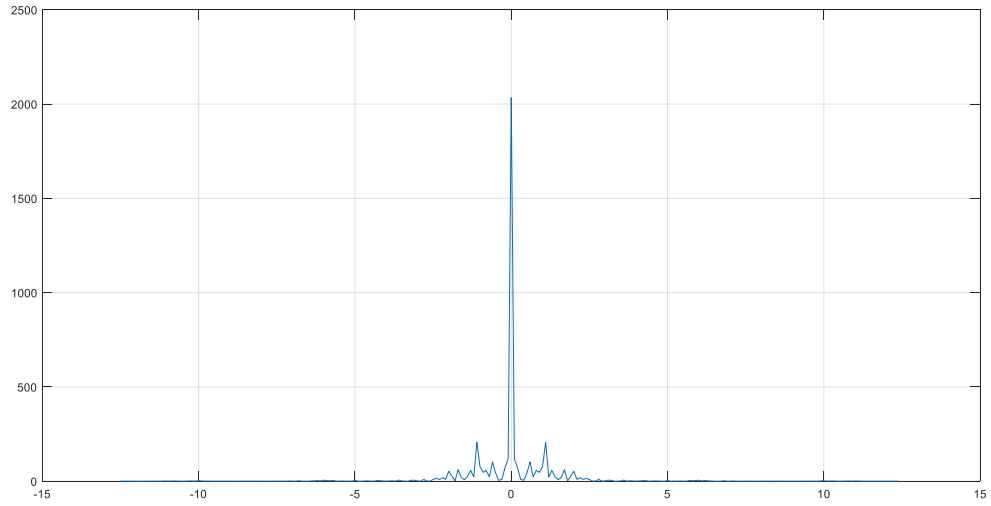


Figure : La DSP de signal S(f) en fonction de la fréquence f.

b)

```
Te = 1/25;
fe = 25;
t = 1.3;
S = 0;
for n = 1:1:length(x)
    S = S + x(n)*(sin(pi*(t-(n-1)*Te)*fe))/(pi*(t-(n-1)*Te)*fe);
end
disp(S)
```

```
>> S = 3.52
```

c) La valeur maximale d'amplitude est 2000 a une fréquence qui égale a 0.

d) Le scripte :

```
vn = zeros(1,500);
vn(2:2:500) = x;
subplot(1,2,1)
plot(1:1:500,vn)
xlabel('Te');
ylabel('Vn');
title('la representation de Vn');
grid on

% Subplot 2
subplot(1,2,2)
plot(1:1:250,x)
xlabel('Te');
ylabel('sn');
title('la representation de Sn');
grid on
```

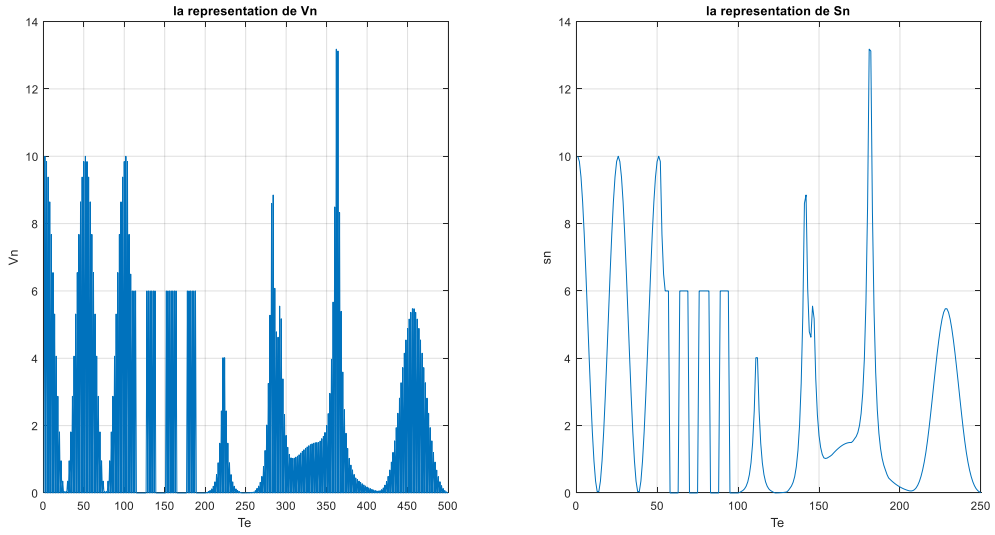


Figure : le signal Vn, et le signal Sn.

e) La fréquence maximale générée dans vn est égale à zéro aussi.

f) Le scripte :

```
% La DSP de Sn
fe = 50;                % sampling frequency
t = 0:(1/fe):(10-1/fe); % time vector
S = vn;
n = length(vn);
X = fft(vn);
figure();%power
fshift = (-n/2:n/2-1)*(fe/n); % zero-centered frequency range
powershift = abs(X).^2/n;
powershift = fftshift(powershift);
plot(fshift,powershift);
grid on

figure();%power
semilogx(fshift, powershift);
title('DSP de sn en fréquences centrées en Hertz');
xlabel('Fréquences (Hz)');
ylabel('DSP');
grid on;
```

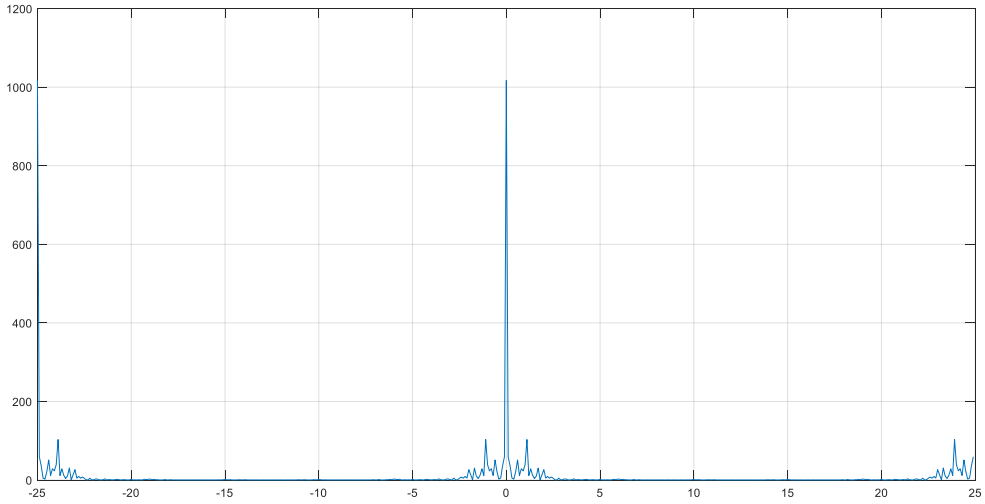


Figure : Le DSP de signal Vn.

Explication des résultats : après l'ajout de valeurs nuls entre les échantillons, ça peut être expliquer comme si on n'a rien fait et c'est pour ça ont eu les mêmes fréquences pour le DSP avec des amplitudes maximales dans les deux cas malgré l'augmentation de la fréquence d'échantillonnage. Pour les valeurs d'amplitude maximale, on a eu presque la moitié de l'amplitude maximale du premier cas, car dans ce cas on prend on considération les valeurs nul et quand on calcule la moyenne on tombe dans la moitié car on a ajouté que des valeurs nulles.

g) L'intérêt d'un filtre passe-bas dans ce cas est le filtrage des valeurs qui sont nuls et les mettre plus lisse et pour avoir un signal sur-échantillonné.
La fréquence de coupure fc doit être inferieur a la fréquence de sur-échantillonnage.
D'une autre façon :
$fréquence\ d'échantillonnage = 25 < fc < 2 * fréquence\ d'échantillonnage = 50\ Hz$

h) $Y_l = x_n$, car on va doubler chaque valeur avec ce filtre et comme ça on aura le même signal de x_n, mais avec une fréquence d'échantillonnage qui est deux fois la fréquence d'échantillonnage du signal originale.

i) La réponse fréquentielle de filtre H1

```
denominator = 1;
numerator = [1,1];
ts = 1/25; % Periode d'échantillonnage
Filter = filt(numerator,denominator,ts)
impulse(Filter)
title('réponse Impulsionelle de H1');
xlabel('Temps(s)');
ylabel('Amplitude');
grid on
figure()
[H,W]=freqz(numerator,denominator,1000);
plot(W,H)
title('DSP de filtre H1');
xlabel('Fréquences');
ylabel('DSP');
grid on
```

```
Filter =

  1 + z^-1

Sample time: 0.04 seconds
Discrete-time transfer function.

Warning: Imaginary parts of complex X and/or Y arguments ignored.
> In TP03 (line 99)
>>
```

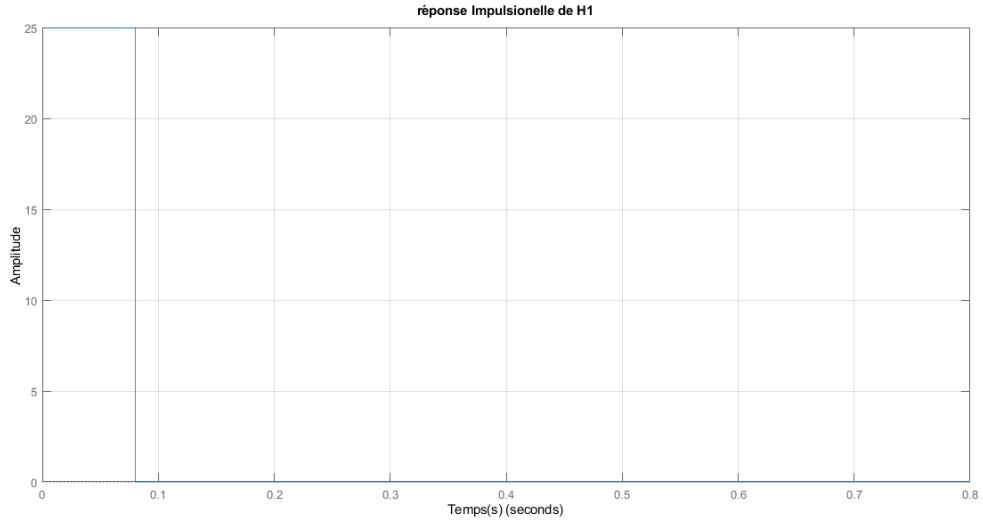


Figure : réponse impulsionnelle de filtre H1.

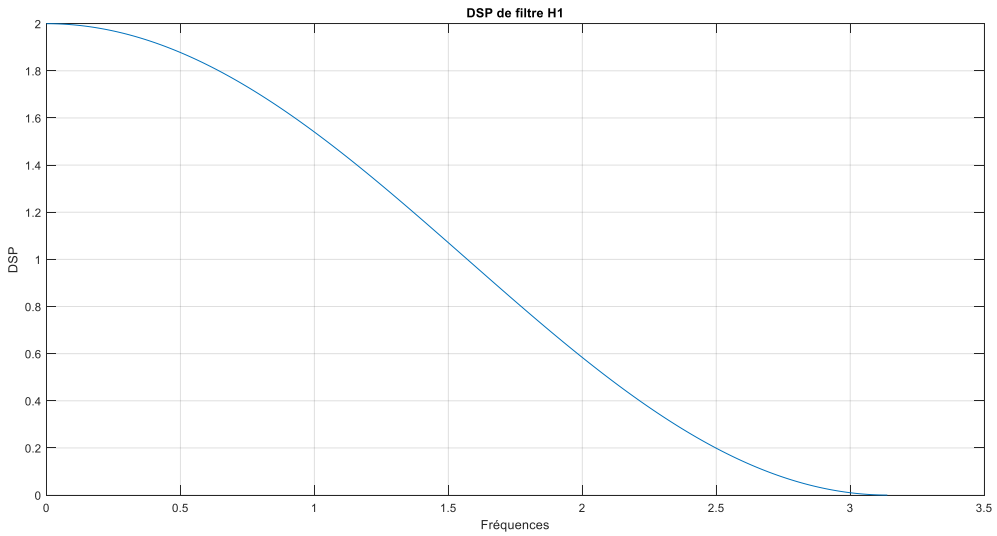


Figure : DSP de filtre H1.

On peut voir que ce filtre est un filtre passe base avec une fréquence de coupure = 1.

j)

```
%Filtrage de signal vn par le filtre H1
y = filter(numerator,denominator,vn);
figure()
plot(t,y)
title('Signal vn Filtré par le filtre H1 ');
```

```
xlabel('Temps(s)');
ylabel('Amplitude');
grid on
```

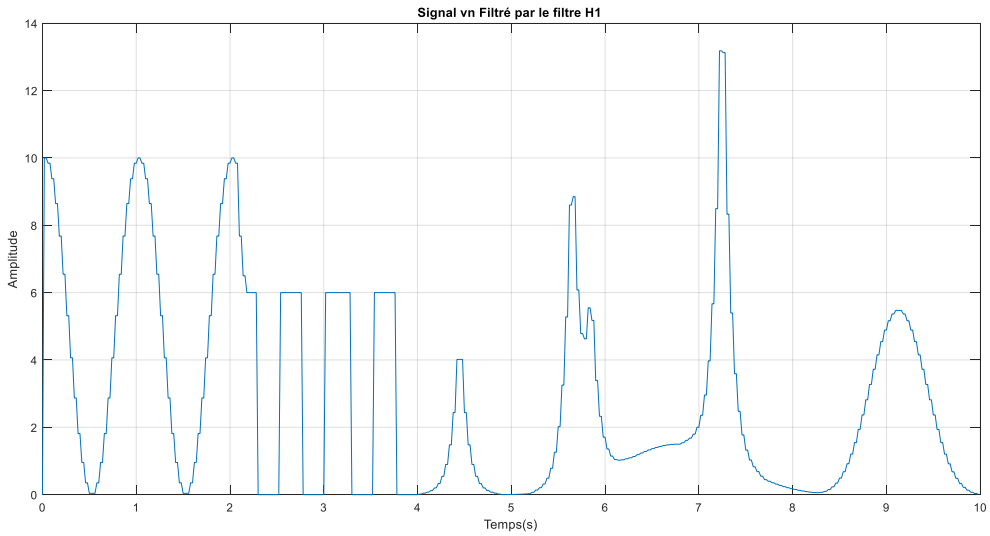


Figure : Signal Vn filtré par le filtre H1.

Le signal obtenu est un signal sur-échantillonné avec le double de la fréquence d'échantillonnage précédente, ainsi on peut voir que ce filtre nous donne deux valeurs pour chaque valeur dans le signal original, d'une autre façon le filtre bloque le signal pour une seul période.

k) $Z_n = s_n \quad si \quad v_n = 0$ Et $Z_n = \frac{1}{2} * (s_n + s_{n-2}) \quad si \quad v_n = s_n$

l)

```
%La réponse fréquencielle de Filtre H2
denominator = 1;
numerator = [1/2,1,1/2];
ts = 1/25; % Periode d'échantillonnage
Filter = filt(numerator,denominator,ts)
impulse(Filter)
title('réponse Impulsionelle de H2');
xlabel('Temps(s)');
ylabel('Amplitude');
grid on
figure()
[H,W]=freqz(numerator,denominator,1000);
plot(W,H)
title('DSP de filtre H2');
xlabel('Fréquences');
ylabel('DSP');
grid on

%Filtrage de signal vn par le filtre H2
y = filter(numerator,denominator,vn);
figure()
plot(t,y)
title('Signal vn Filtré par le filtre H2 ');
xlabel('Temps(s)');
ylabel('Amplitude');
grid on
```

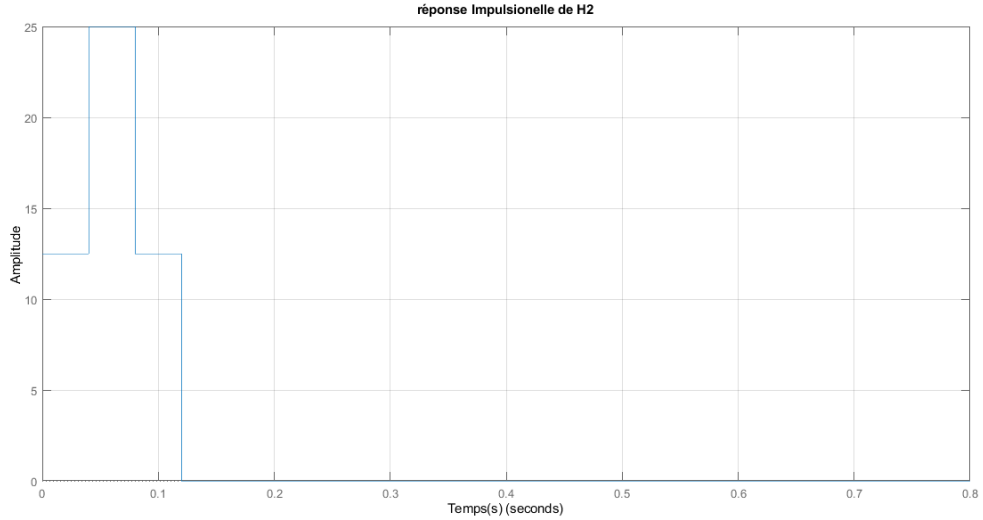
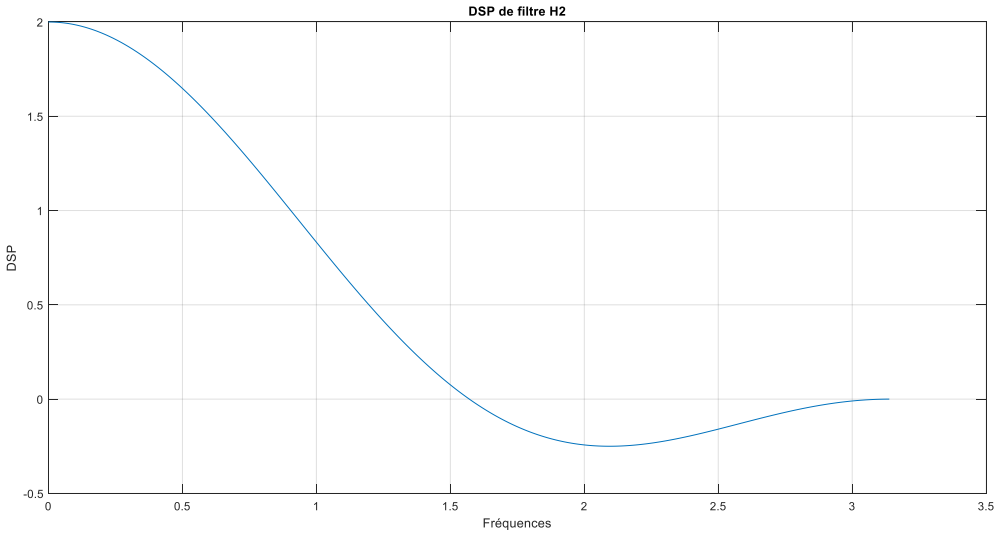


Figure : réponse impulsionnelle de filtre H2



Filtre : DSP de filtre H2.

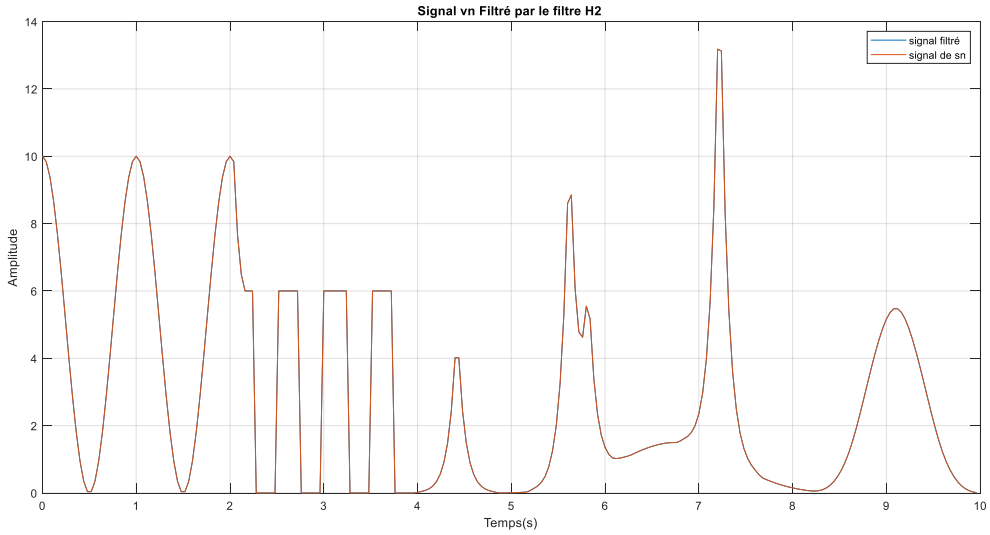


Figure : Signal filtré par le filtre H2 avec le signal original.

On constate qu'on a obtenu un signal qui est à peu près identiques au signal original et avec une fréquence d'échantillonnage plus élevé. Ce signal est mieux que le signal obtenu par le filtre H1 (précèdent).

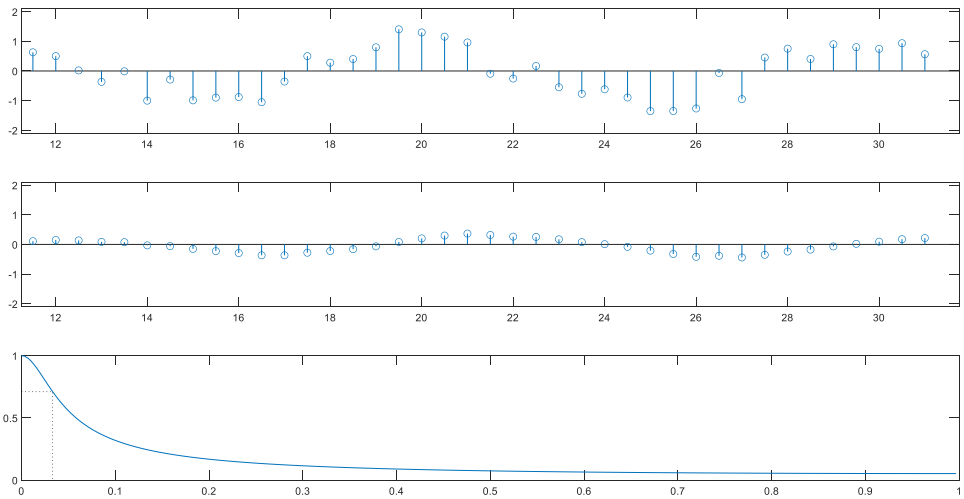
On peut conclure que c'est possible de sur-échantillonner un signal en ajoutant des zéros et on utilise un filtre passe-bas.

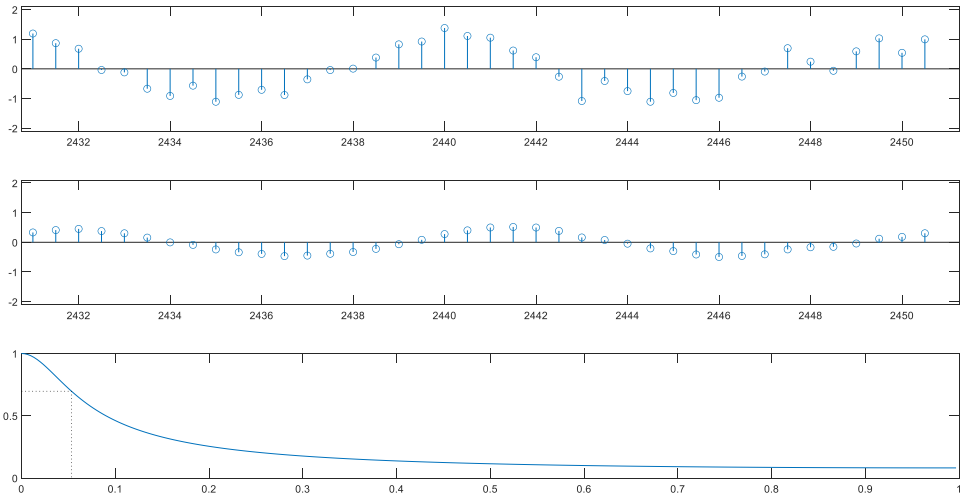
g. Pour ce cas le filtre idéal est de fréquence de coupure égale à la fréquence d'échantillonnage du signal original 25Hz, ainsi de gabarit idéal est un signal DSP carré qui coupe l'axe des fréquences à la fréquence de coupure.

Séance 04 :

IV. Filtrage d'un signal bruité :

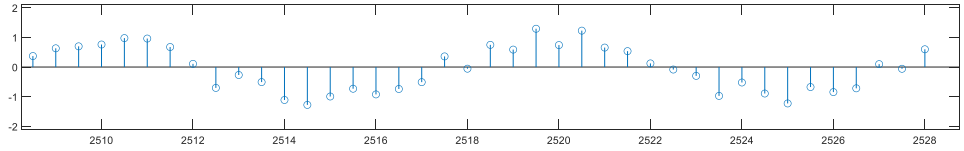
I. .

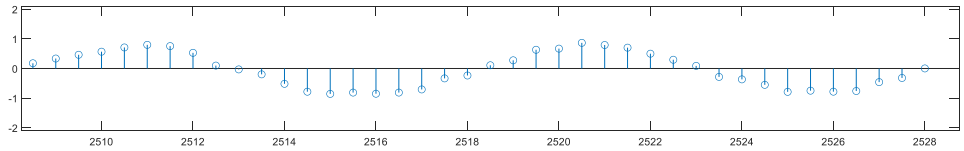


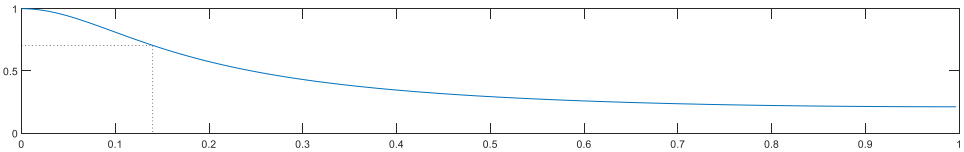

2
1
0
-1
-2
2432
2434
2436
2438
2440
2442
2444
2446
2448
2450

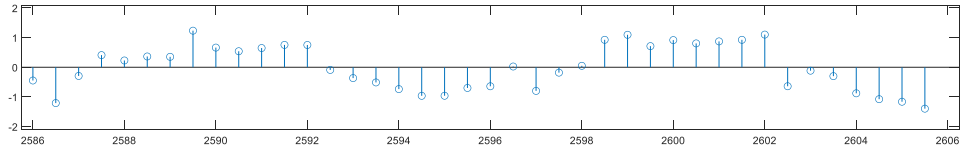
2
1
0
-1
-2
2432
2434
2436
2438
2440
2442
2444
2446
2448
2450

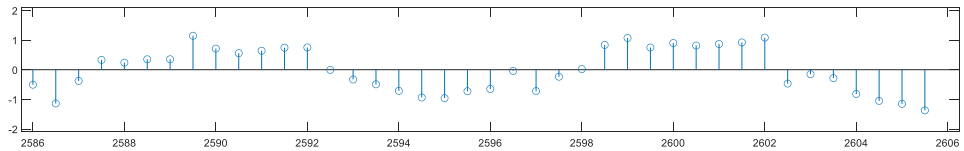
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

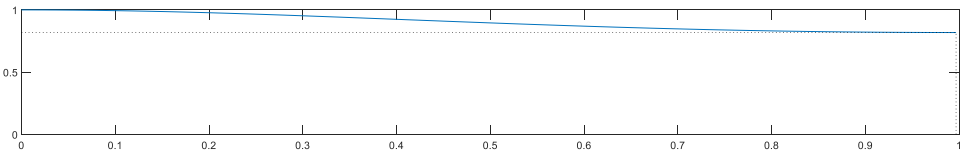

2
1
0
-1
-2
2510
2512
2514
2516
2518
2520
2522
2524
2526
2528


2
1
0
-1
-2
2510
2512
2514
2516
2518
2520
2522
2524
2526
2528


1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


2
1
0
-1
-2
2586
2588
2590
2592
2594
2596
2598
2600
2602
2604
2606


2
1
0
-1
-2
2586
2588
2590
2592
2594
2596
2598
2600
2602
2604
2606


1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

m)

Le rapport signal-à-bruit (RSB) est une mesure de la qualité d'un signal en comparaison avec le niveau de bruit qui l'entoure. Il est défini comme le rapport de la puissance du signal (S) à la puissance du bruit (N). Mathématiquement, cela s'exprime souvent en termes de puissances :

$$RSB = \frac{P_S}{P_N}$$

Où P_S est la puissance du signal et P_N est la puissance du bruit.

Pour un signal discret $x[n]$, la puissance P_S peut être calculée comme suit :

$$P_S = \frac{1}{N}\sum_{n=0}^{N-1}|x[n]|^2$$

Où N est le nombre d'échantillons dans le signal.

Pour obtenir le RSB en décibels (dB), on utilise la formule suivante :

$$RSB_{dB} = 10 \cdot log_{10}(RSB)$$

Cela convertit le rapport en une échelle logarithmique, plus pratique pour exprimer de grandes variations de puissance.

n) Les signaux bruités :

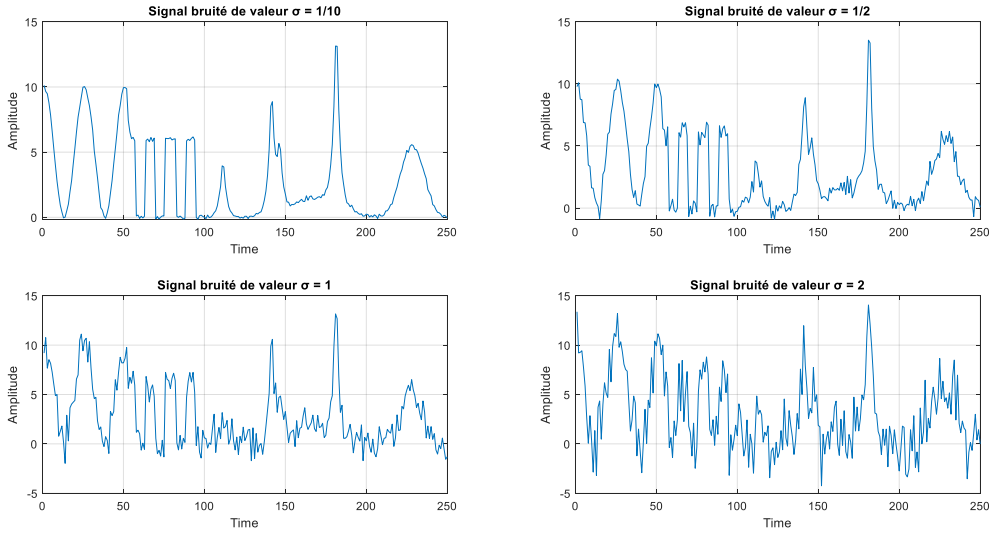


Figure : Signaux bruités avec différentes variances.

Le rapport signal bruit pour chaque signal :

Le rapport signal a bruit pour (σ : 1/10) = 32.7300

Le rapport signal a bruit pour (σ : 1/2) = 19.0356

Le rapport signal a bruit pour (σ : 1) = 12.0098

Le rapport signal a bruit pour (σ : 2) = 6.4609

On remarque que le rapport signal bruit diminue lorsque la valeur l'écart type augmente. Ce qui est logique car la puissance de signal de bruit augmente, lorsque on augmente l'écart type.

o)

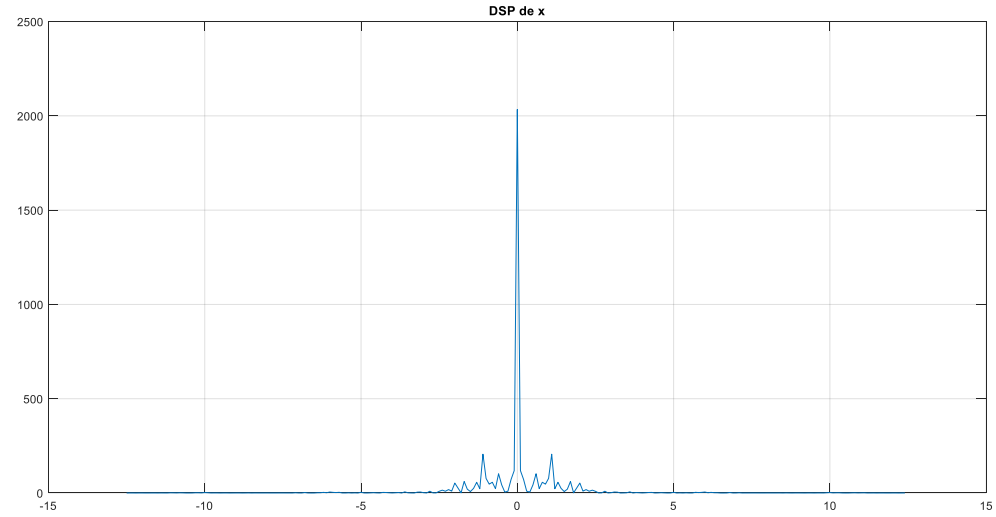


Figure : DSP de signal originale sans bruit.

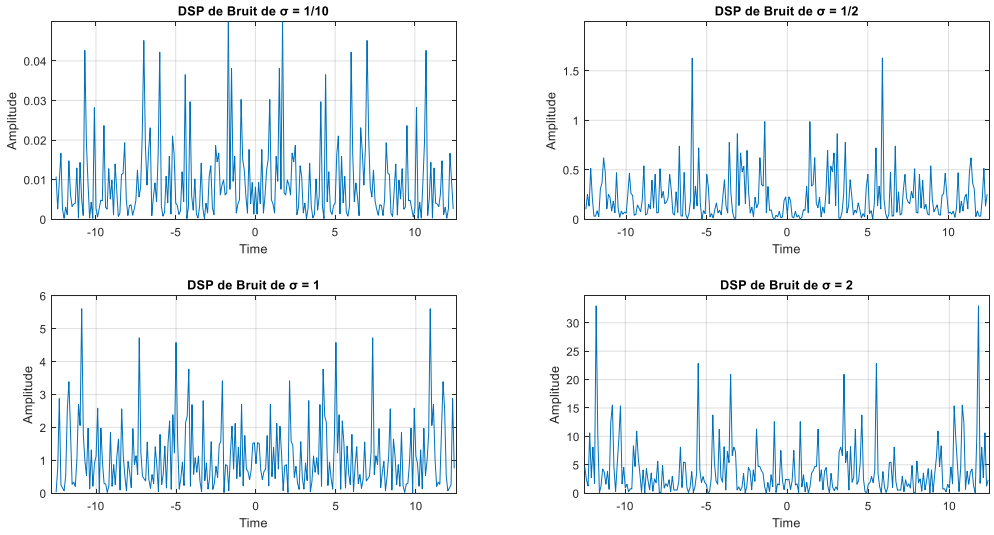


Figure : DSP des signaux bruités avec différentes variances.

On remarque qu'on a des amplitudes importantes pour les grandes fréquences dans le DSP de bruit. Donc le rôle de filtre c'est d'éliminer les amplitudes pour les grandes fréquences.

p) La sortie Z_n du filtrage de y par le filtre H :

$$Z_n = \frac{1}{p} \cdot y_n + \frac{1}{p} \cdot y_n \cdot z^{-1} + \frac{1}{p} \cdot y_n \cdot z^{-2} + \frac{1}{p} \cdot y_n \cdot z^{-3} + \frac{1}{p} \cdot y_n \cdot z^{-4} \dots\dots\dots \frac{1}{p} \cdot y_n \cdot z^{-p}$$

q) L'application de filtre 1/p :

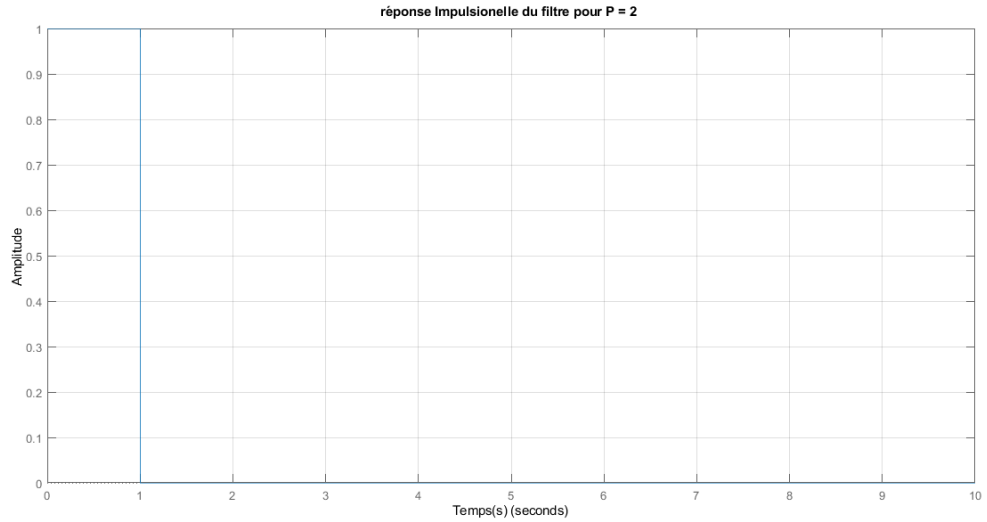


Figure : réponse impulsionnelle du filtre pou P = 2.

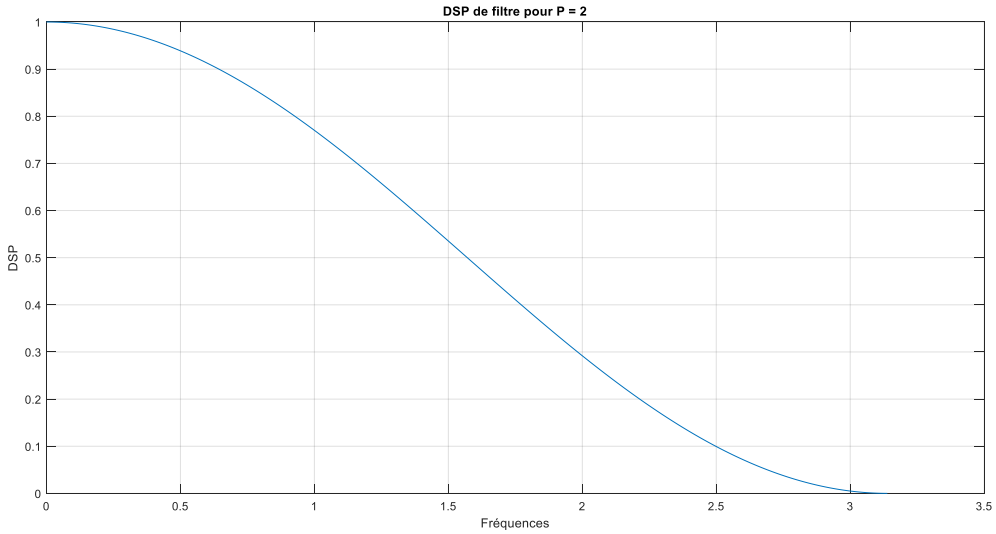


Figure : DSP de filtre pour P=2.

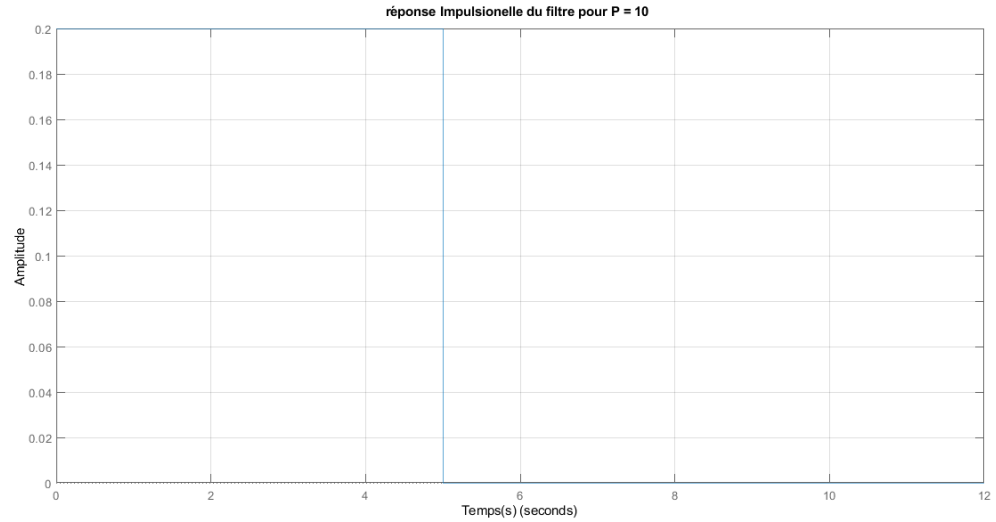


Figure : réponse impulsionnelle du filtre pour P = 10.

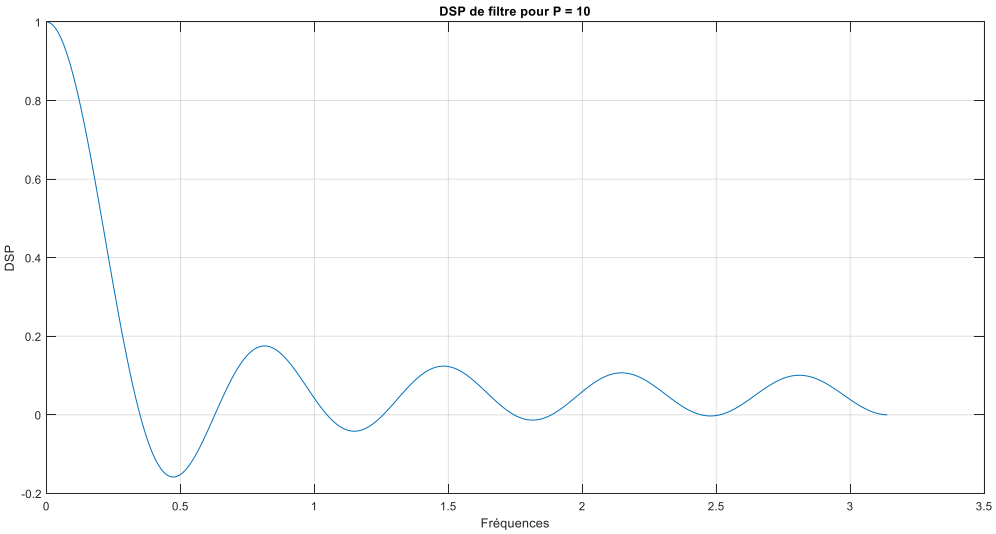


Figure : DSP de filtre pour P=10.

Pour P = 2 la réponse filtrée

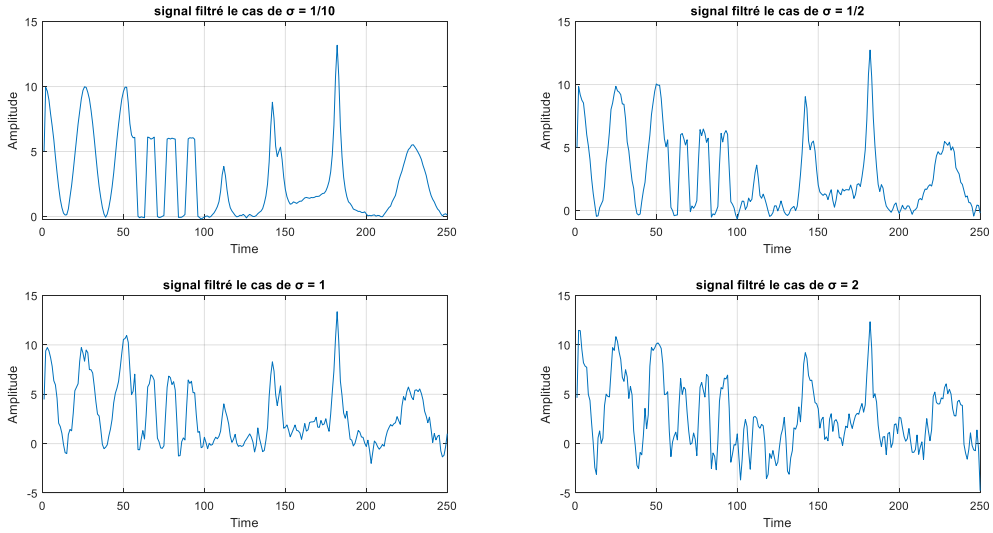


Figure : signaux filtrés pour différents cas de variances.

Pour P = 10 la réponse filtrée

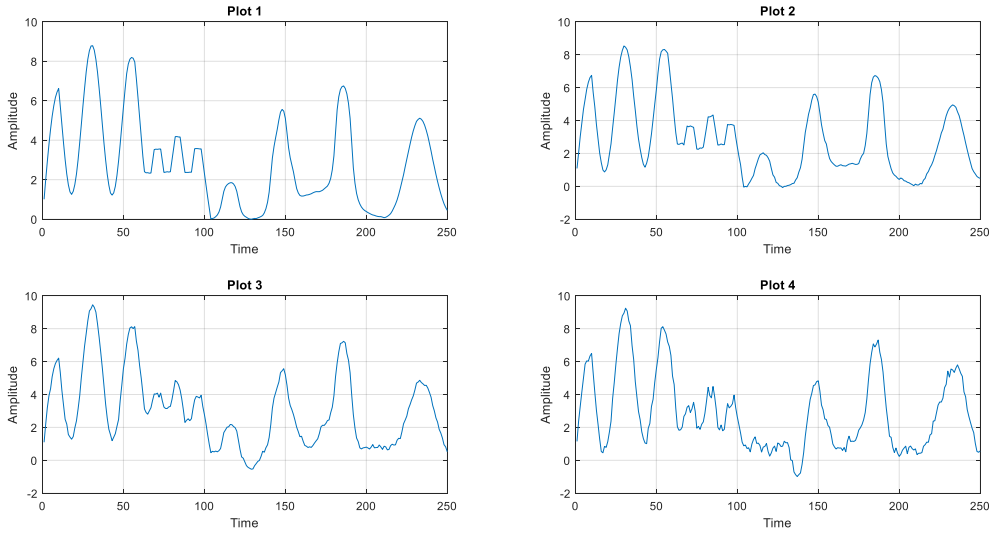


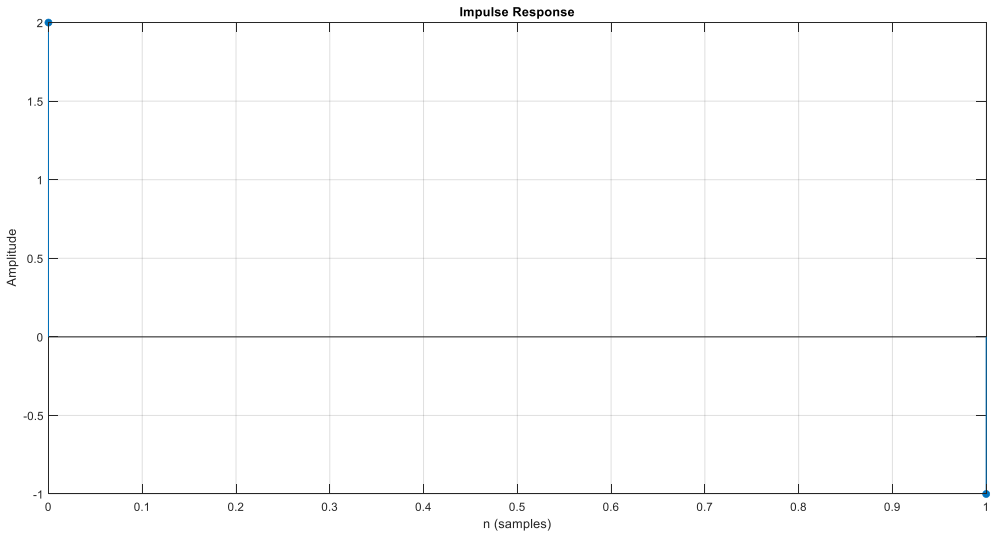
Figure : signaux filtrés pour différents cas de variances.

On peut conclure que lorsque augmente la valeur de p, Le filtre élimine plus les hautes fréquences et ça on peut le voir dans la représentation de DSP des deux filtres, donc on doit choisir un P qui élimine les bruits et au même temps garde les fréquences du signal original.

Le problème avec ça, est que l'augmentation de p nécessite plus de calcule numérique.

15. Si la réponse impulsionnelle est finie pour tout n, alors le filtre est un filtre RII (notre cas). Si la réponse impulsionnelle n'est pas finie, le filtre est un filtre RIF (Réponse à Impulsion Finie).

17. Vérification des coefficients de filtre a1 et a2



18. La condition pour assurer la stabilité :

$$z = \left|\frac{a_2}{a_1}\right| < 1$$

19. La condition est pour garantir que le filtre n'est pas un amplificateur ou abaisseur, (il a un gain statique qui égale a 1). Cela signifie que la somme de tous les échantillons de la réponse impulsionnelle est égale à 1. Et cela implique que le filtre conserve l'énergie du signal d'entrée.

20.

$$a_2 + a_1 = 1$$

$$\left|\frac{a_2}{a_1}\right| < 1$$

On peut choisir a2 = 0.001 et a1 = 0.999

22. Signal bruité et signal filtré :

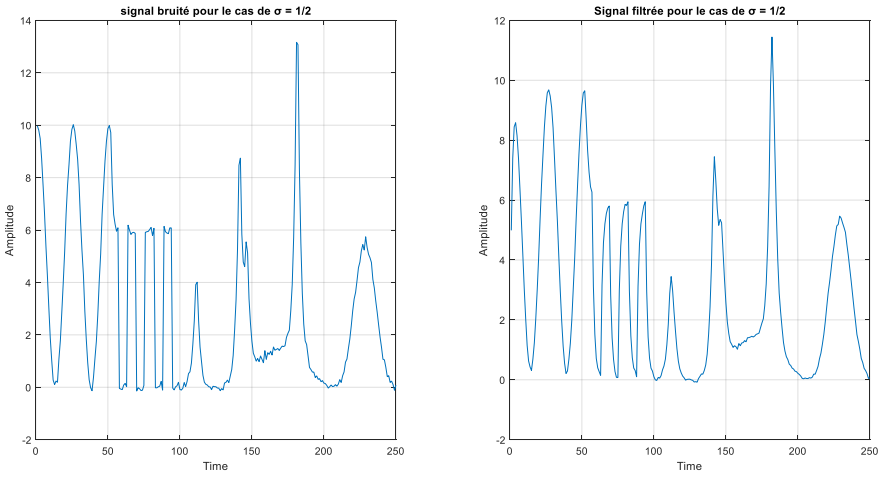


Figure : Signal bruité et signal filtrée.

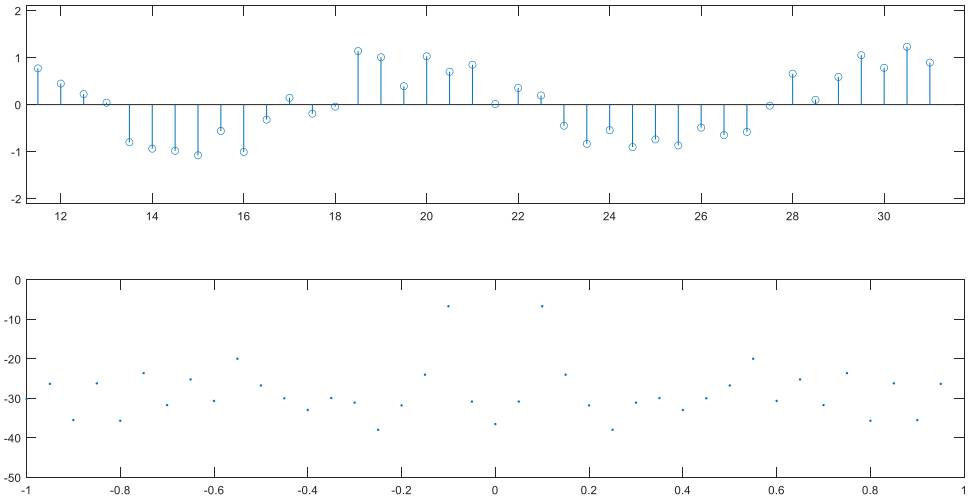
On constate que le signal filtré n'a pas de bruit important car le filtre a éliminé ces bruits, mais on peut voir que le signal devient lisse ce qui implique qu'on a perdu un peu de d'information au niveau de signal original.

23. La premier méthode avec le filtre moyenneur, a donné des résultats acceptables mais avec un ordre supérieur, et ça nécessite une puissance de calcul importante. Pour la deuxième méthode avec le filtre RII, on a eu des bons résultats seulement avec une détermination de paramètres de filtre et ainsi dans cette méthode on n'a pas un calcule important.

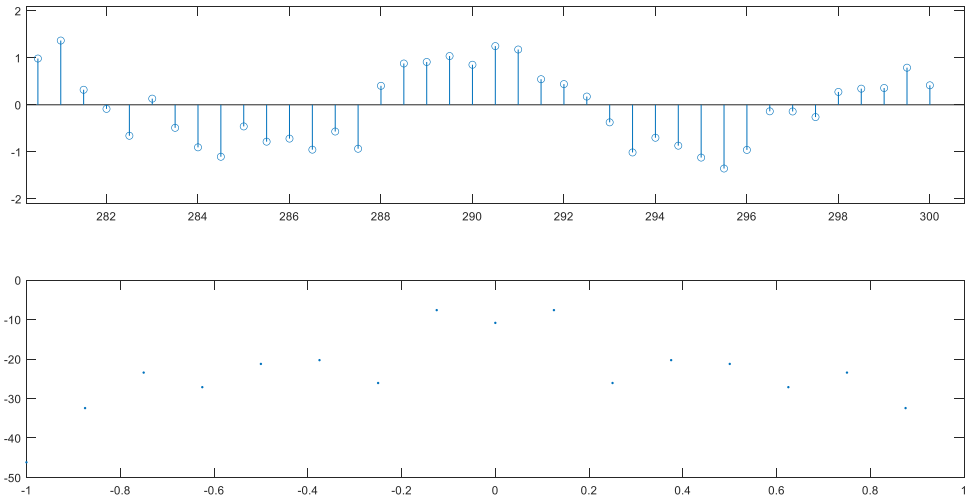
Séance 05 :

V. Synthèse de filtres :

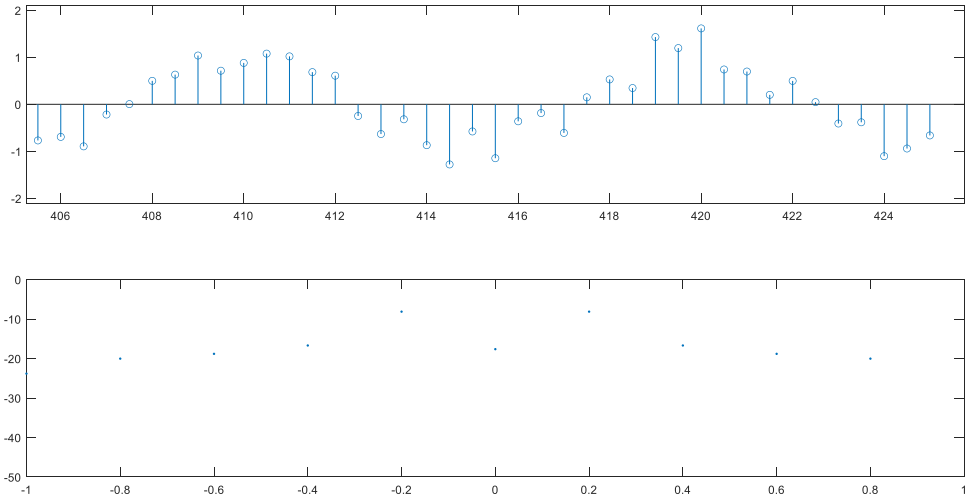
- Pour $delta\_t = 20$:



- Pour $delta\_t = 8$:



- Pour $delta\_t = 5$



VI. Filtre passe-bas :

7.

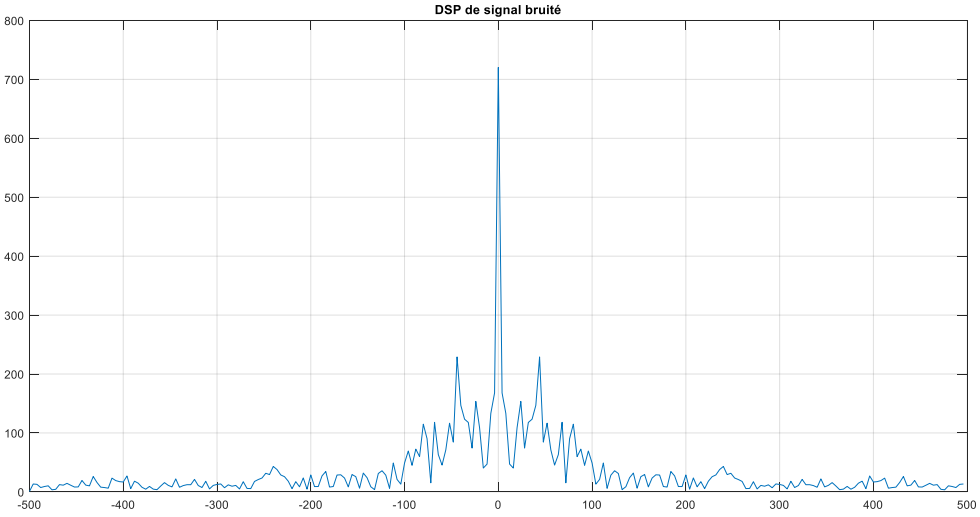


Figure : Le DSP du signal bruité.

- Pour $fc = 50$:

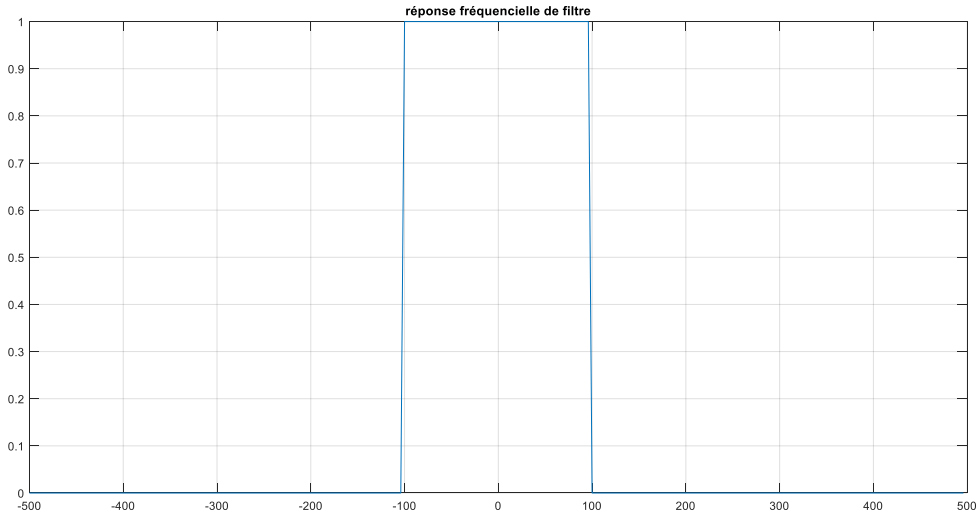


Figure : La réponse fréquentielle du filtre avec $fc = 50$.

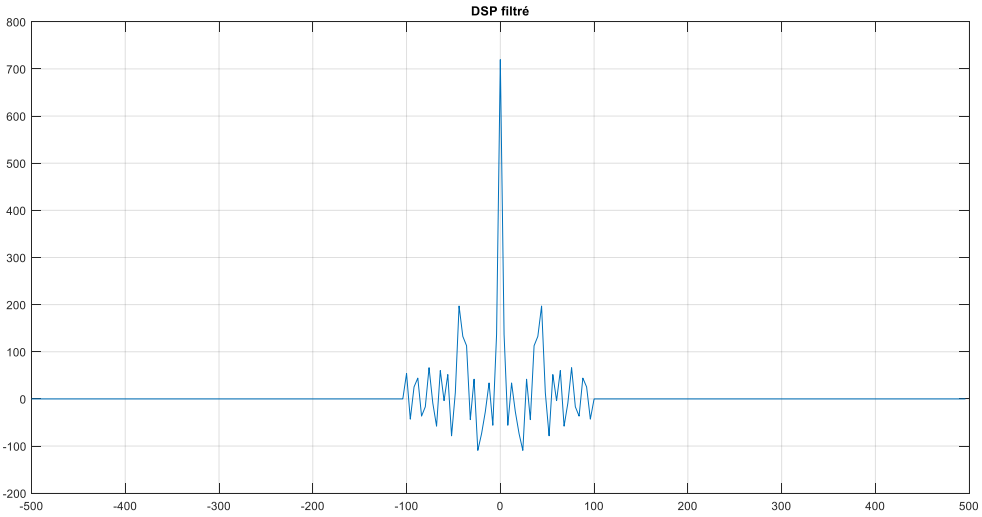


Figure : La réponse fréquentielle filtrée.

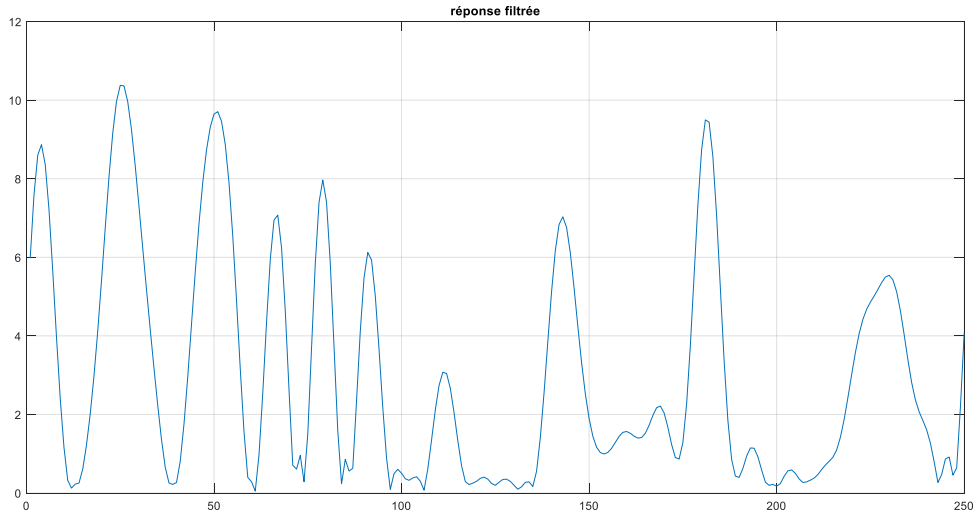


Figure : Réponse filtrée.

- Pour $fc = 10$

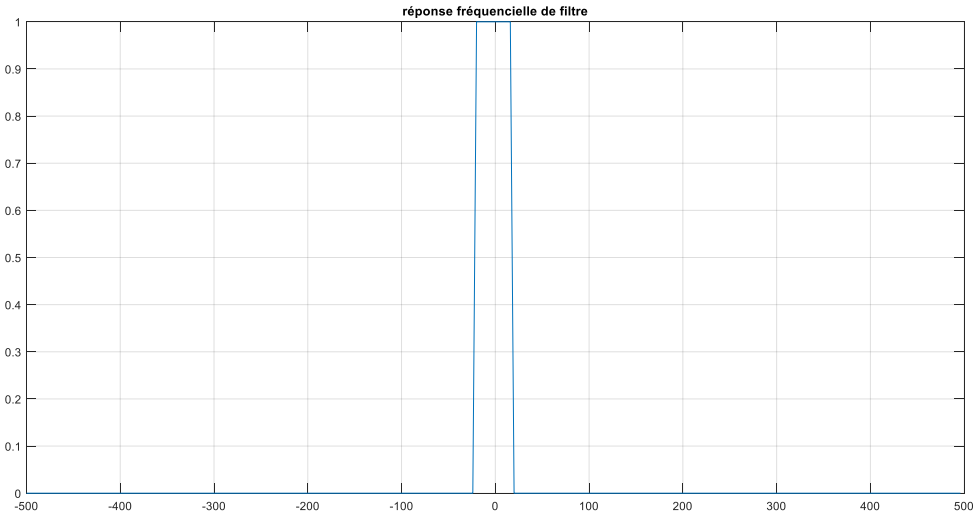


Figure : La réponse fréquentielle du filtre avec $fc = 10$.

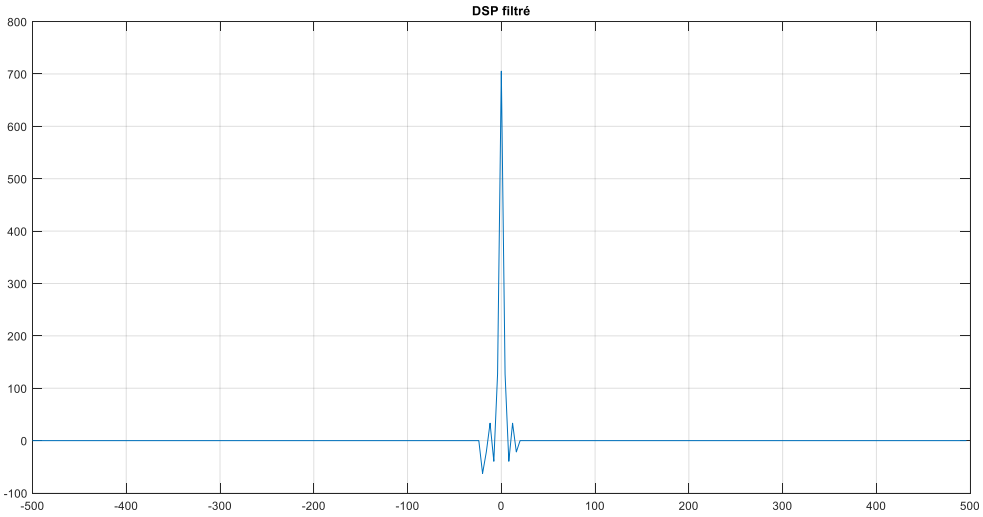


Figure : La réponse fréquentielle filtrée.

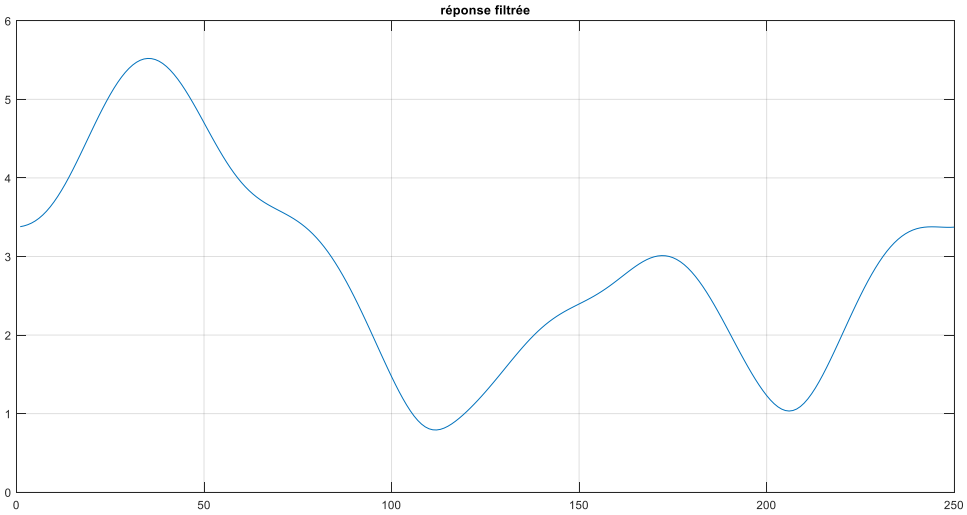


Figure : Réponse filtrée.

Commentaire : pour $f_c = 50Hz$, ça nous donne un signal bien filtré mieux que la valeur $f_c = 10Hz$ qui élimine les fréquences liées au signal originale. Donc on peut voir qu'à chaque fois on diminue la fréquence, les signaux de fréquences plus grande que la fréquence de coupure vont être à peu près éliminés, donc on perd plus d'information sur notre signal original avec l'élimination de bruit.

8. Le script :

```
%Definition de la reponse frequentielle de signal x
fe = 1000;                % sampling frequency
y2 = x + 0.5*randn(1,length(x));
fc = 50;
frequence_normalise = fc/(fe/2);
% Question 07 et 08
B = fir1(1,frequence_normalise,'low');
figure()
freqz(B,1,1000,fe)
signal = y2;
```

```
filtered_signal = filter(B, 1, signal);
% Visualisation du signal d'origine et du signal filtré
figure();
subplot(2,1,1);
plot(signal);
title('Signal avec bruit (20 Hz) + Bruit aléatoire');
xlabel('Temps');
ylabel('Amplitude');
grid on
subplot(2,1,2);
plot(filtered_signal);
title('Signal filtré (fréquence de coupure à 50 Hz)');
xlabel('Temps');
ylabel('Amplitude');
grid on
```

Pour un filtre « fir1 » d'ordre $= 1$ avec une fréquence de coupure de $50Hz$:

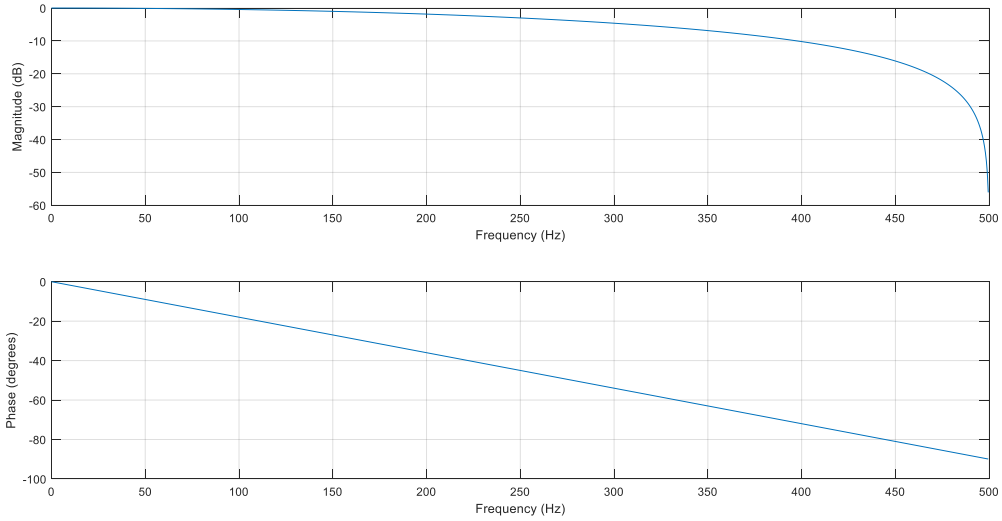


Figure : Diagramme de Bode pour le filtre d'ordre 1 et de fréquence de coupure : $50Hz$.

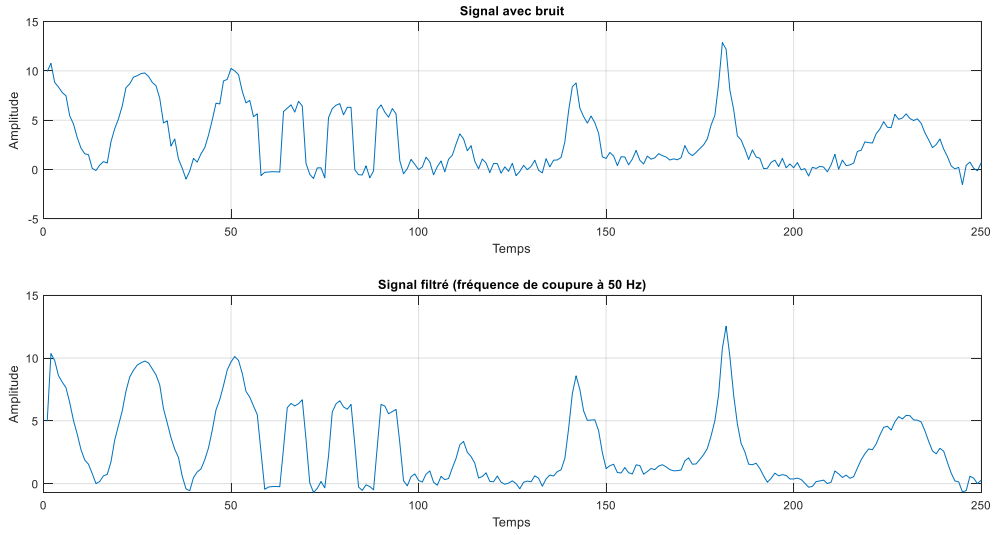


Figure : Le signal bruité (en haut), signal filtré (en bas).

Pour un filtre « fir1 » d'ordre $=$ 10 avec une fréquence de coupure de $50Hz$:

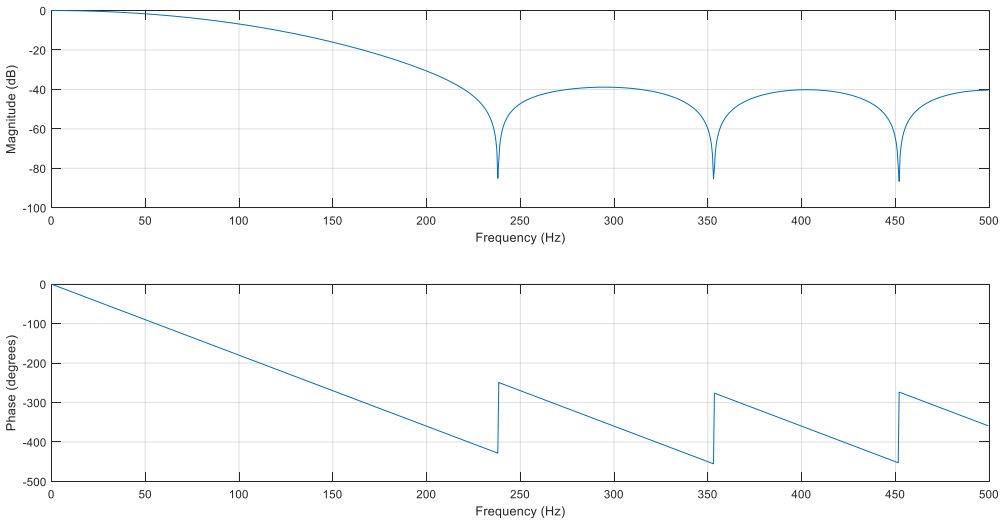


Figure : Diagramme de Bode pour le filtre d'ordre 10 et de fréquence de coupure : $50Hz$.

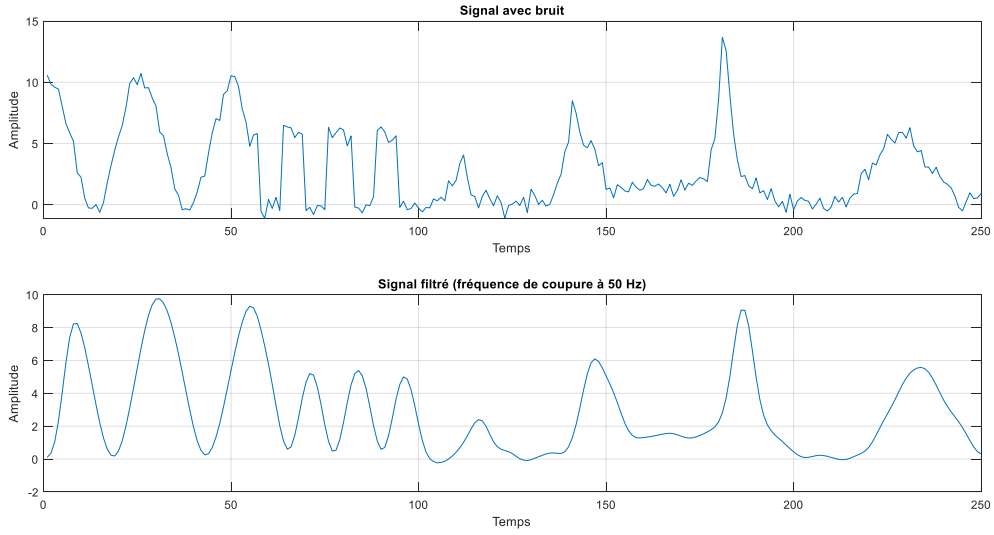


Figure : Le signal bruité (en haut), signal filtré (en bas).

Pour un filtre « fir1 » d'ordre $= 1$ avec une fréquence de coupure de $10Hz$:

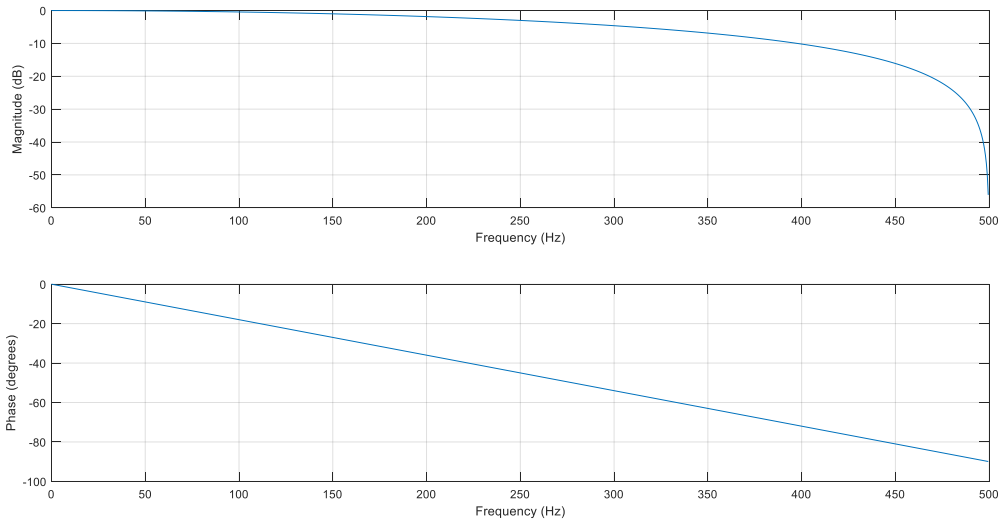


Figure : Diagramme de Bode pour le filtre d'ordre 1 et de fréquence de coupure : $10Hz$.

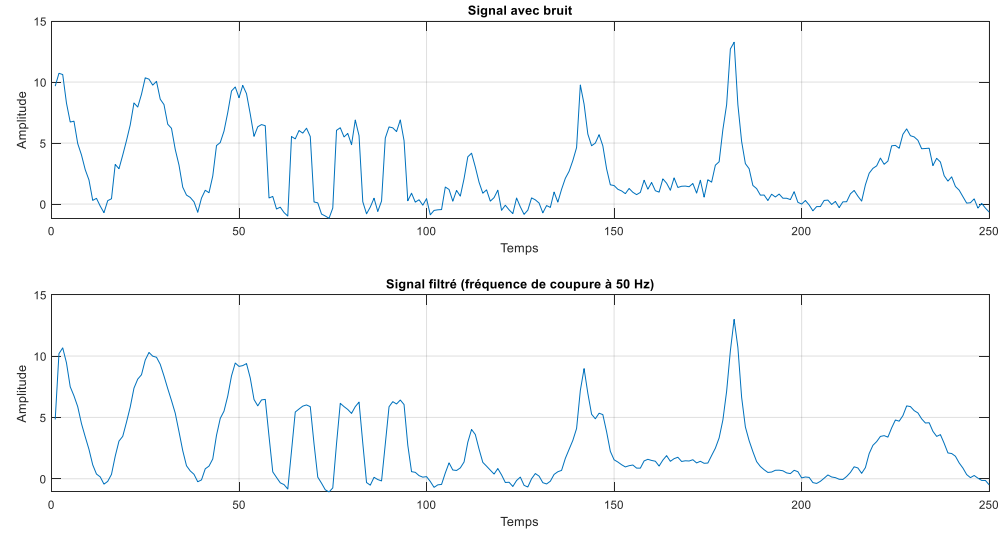


Figure : Le signal bruité (en haut), signal filtré (en bas).

9. L'effet de changement la fenêtre : à chaque fois on diminue la fenêtre on aura moins de bruit mais si on choisit une fenêtre très petite, on perdre des fréquences dans le signal originale.
10. Le calcul de l'ordre optimal :

```
% Spécifications du filtre
Rp = 3;  % Gain en dB de la partie passante
Rs = 15;  % Gain en dB de la partie non-passante
fc = 50;  % Fréquence de coupure en Hz
fa = 60;  % Fréquence d'atténuation en Hz
% Calcul de l'ordre optimal du filtre
[N, Wn] = buttord(fc, fa, Rp, Rs, 's');
disp(['Ordre optimal du filtre : ', num2str(N)]);
disp(['Fréquence de coupure normalisée : ', num2str(Wn)]);
```

```
>> seance05_partie02
```

```
Ordre optimal du filtre : 10
Fréquence de coupure normalisée : 50.5649
```

VII. Filtre coupe-bande :

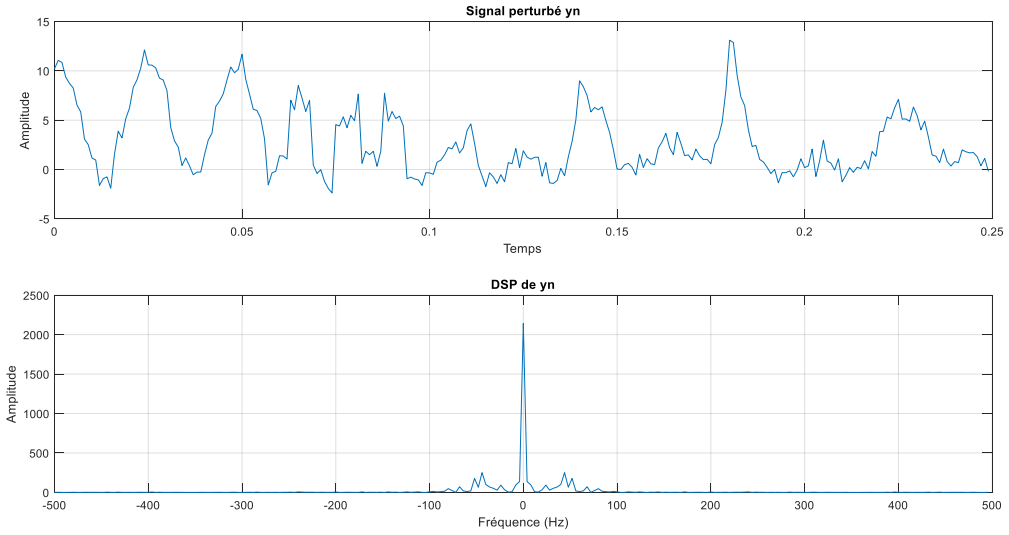


Figure : Signal perturbé (Haut), Son DSP (Bas).

15. Commentaire : On constate que les fréquences de haute amplitude sont autours de zéro. Mais on peut voir aussi deux pics en 50Hz et -50Hz due au signal sinusoïde ajouté et d'autre avec des grandes fréquences due aux bruits appliqué sur ce signal.

16.

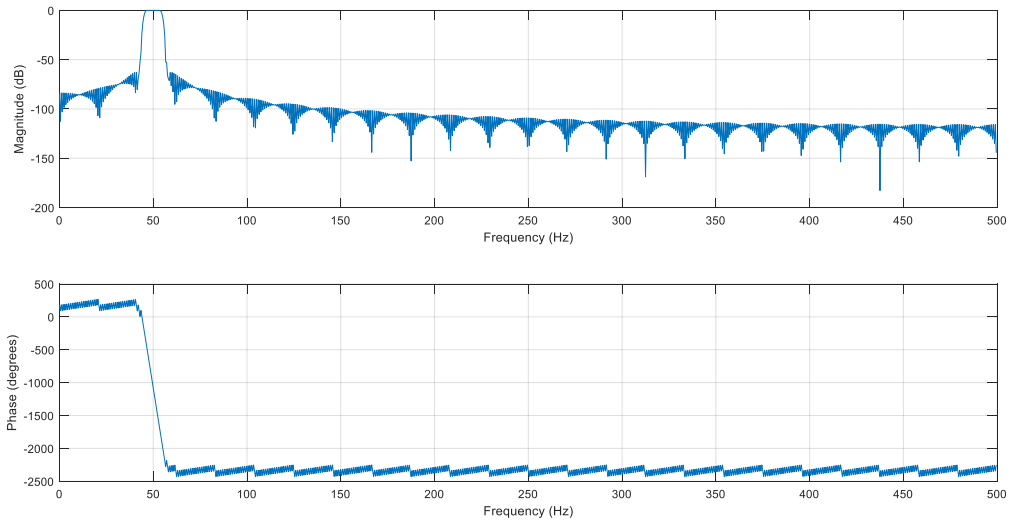


Figure : Diagramme de Bode pour le Filtre RIF.

On constate dans la figure la réponse fréquentielle d'un filtre RIF. On peut voir qu'au point 50 on a une coupure ce qui corresponds à la fréquence du signal sinusoïde.

Pour un filtre RIF (réponse impulsionnelle infini), il faut un grand nombre de coefficients et dans notre cas on a choisi 100 coefficients pour le numérateur. Et pour le dénominateur on a seulement un.

17.

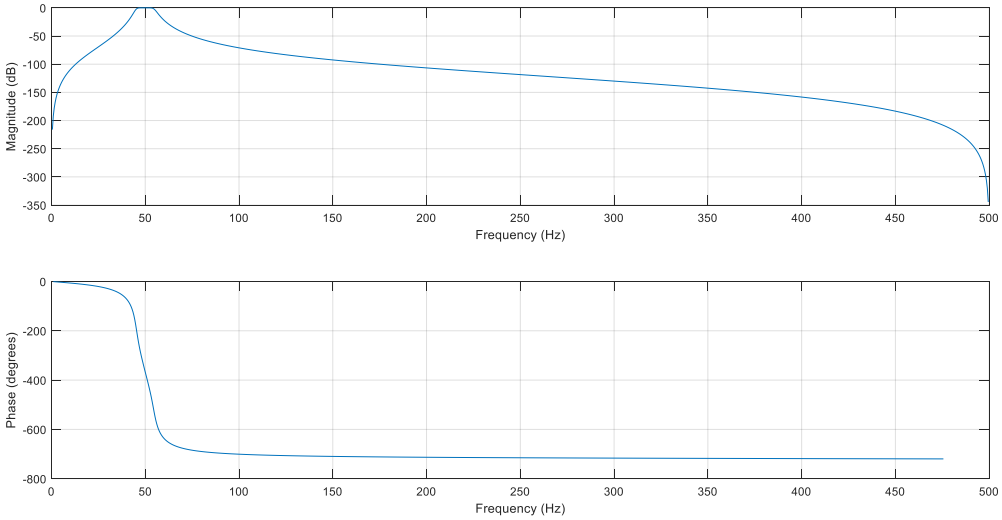


Figure : Diagramme de Bode pour le Filtre RII.

On constate dans la figure la réponse fréquentielle d'un filtre RII. On peut voir qu'au point 50 on a une coupure ce qui corresponds à la fréquence du signal sinusoïde.

```
>> b

b =

   1.0e-05 *

    0.0898         0   -0.3594         0    0.5391         0   -0.3594         0    0.0898

>> a

a =

    1.0000   -7.4560   24.6857  -47.3727   57.6139  -45.4670   22.7396   -6.5919    0.8486
```

On prend six coefficient pour le numérateur et le dénominateur.

18. Le filtrage :

Le script:

```
clear all
close all
clc
% Paramètres du signal
Fs = 1000; % Fréquence d'échantillonnage
t = 0:1/Fs:249/1000; % Vecteur temps
fe = Fs
% Signal utile perturbé par la sinusoïde
xn = load('signalbase.mat').x;

% Signal sinusoïdal parasite
t_sin = t; % Vecteur temps pour la sinusoïde
f_sin = 50; % Fréquence de la sinusoïde
A_sin = 1.2; % Amplitude de la sinusoïde
sinusoid = A_sin * sin(2*pi*f_sin*t_sin);


% Ajout du bruit blanc centré de variance 0.5
bn = sqrt(0.5) * randn(size(t)); % Génération du bruit blanc
y2 = xn + bn + sinusoid; % Signal total perturbé


fc = 250;
frequence_normalise = fc/(fe/2);
% Question 07 et 08
B = fir1(1,frequence_normalise,'low');
figure()
freqz(B,1,1000,fe)
signal = y2;
filtered_signal = filter(B, 1, signal);




% Spécifications du filtre
f0 = 50; % Fréquence caractéristique
bw = 10; % Largeur de bande
fs = 1000; % Fréquence d'échantillonnage

% Calcul des fréquences de coupure normalisées
f1 = (f0 - bw/2) / (fs/2); % Fréquence de coupure inférieure normalisée
f2 = (f0 + bw/2) / (fs/2); % Fréquence de coupure supérieure normalisée

% Création du filtre coupe-bande
order = 2; % Ordre du filtre (par exemple)

[b, a] = butter(order, [f1, f2], 'stop'); % Conception du filtre
Butterworth

% Analyse de la réponse en fréquence
figure()
freqz(b, a);
```

```
y_filtre_cb_rif = filter(b, a, filtered_signal);


% Affichage des signaux filtrés
figure()
subplot(4,1,1);
plot(xn);
title("Signal d'origine");


subplot(4,1,2);
plot(y2);
title("Signal bruité");

subplot(4,1,3);
plot(filtered_signal);
title('Signal filtré par passe-bas');

subplot(4,1,4);
plot(y_filtre_cb_rif);
title('Signal filtré par passe-bande RIF');
```

J'ai utilisé un filtre passe bas de fréquence de coupure = 100Hz :

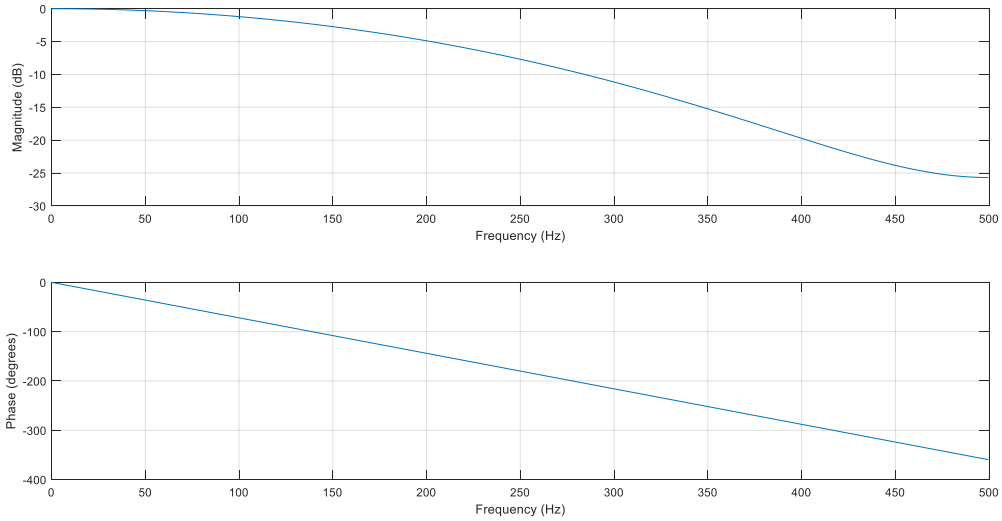


Figure : Diagramme de Bode pour le Filtre passe-bas.

Puis, j'ai utilisé un filtre passe-bande de largeur de bande = 10 :

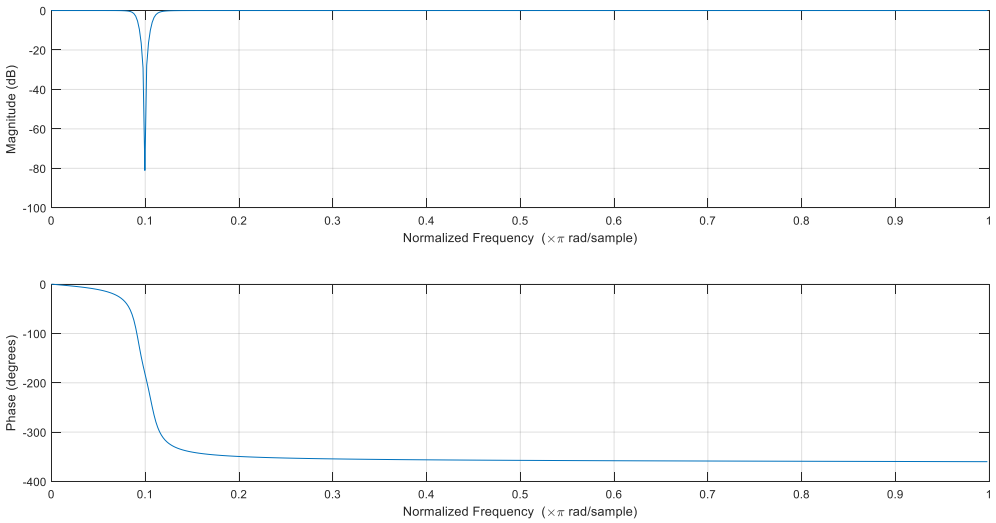


Figure : Diagramme de Bode pour le Filtre passe-bande.

Résultat obtenu est le suivant :

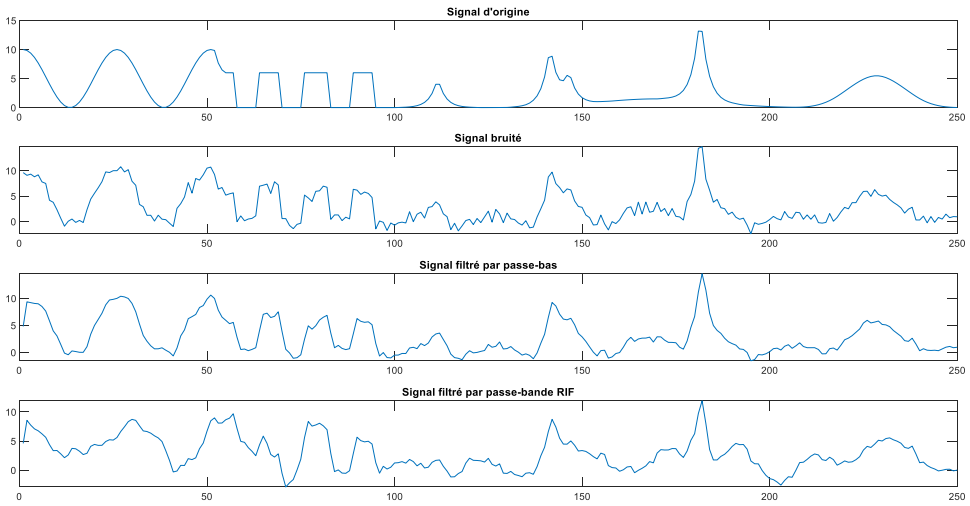


Figure : signaux filtrés.

- On doit utiliser un filtre passe bas pour éliminer les grandes fréquences de bruit et puis un filtre passe bande pour éliminer les pics dans le DSP pour le signal sinusoïdale, comme ça on aura le signal filtré avec un DSP sans amplitudes pour les hautes fréquences de bruit et pour les deux fréquences de signal sinusoïdal.

19.

Pout une largeur de bande de 5 on aura :

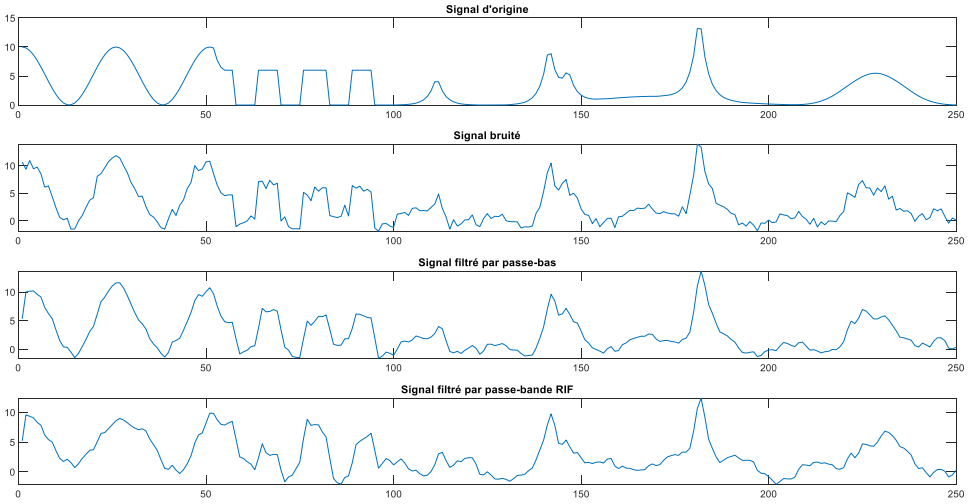


Figure : signaux filtrés.

Pour une largeur de bande de 1 :

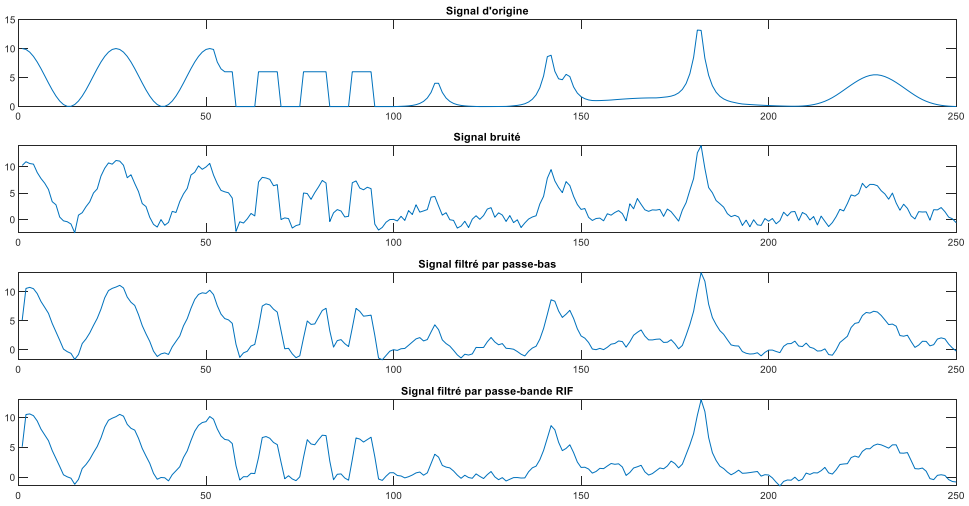


Figure : signaux filtrés.

Pour une largeur de bande = 40 :

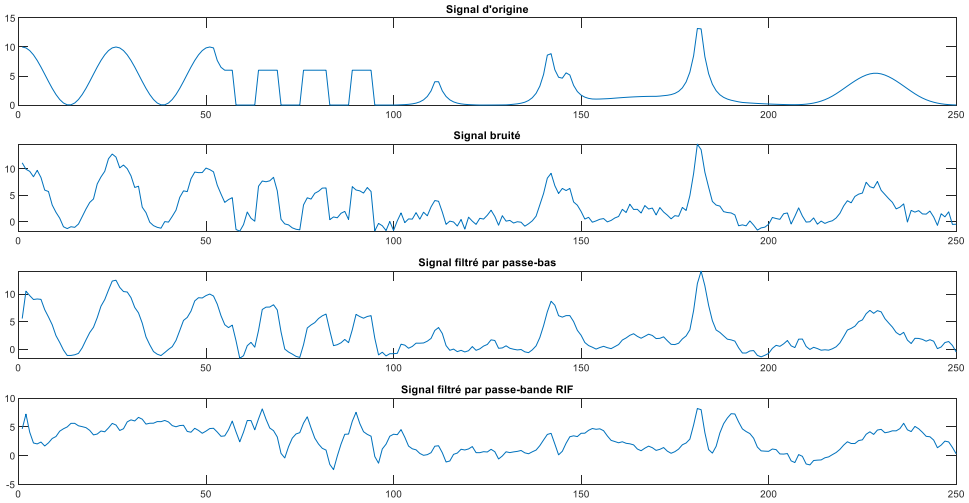


Figure : signaux filtrés.

On constate qu'à chaque fois on diminue la largeur de bande on aura un signal plus proche au signal original, et si on augmente la largeur de bande on élimine les sinusoïdes mais aussi avec quelques fréquences porteuse des informations du signal original. Donc en conclusion une largeur minimale autour de la fréquence d'un signal sinusoïdale peut nous donner un bon filtrage de bruit sinusoïde.