Jean-Marc DUFOUR
Adil BENNIS

Mars 2007
P2009

EE 345 – Traitement du Signal

TP 1 – Approche du traitement numérique du signal

L'objectif de ce travail est d'aborder la manipulation de signaux discrets en utilisant le logiciel MATLAB.

Sommaire

Introduction

Ce TP est l'occasion de se familiariser avec le logiciel MATLAB au traitement du signal via des exemples simples : représentation de signaux en temps et en fréquence, application des théorèmes de base du traitement numérique du signal (échantillonnage de Shannon), et exemples concrets (signaux DTMF et générateur sinusoïdaux).

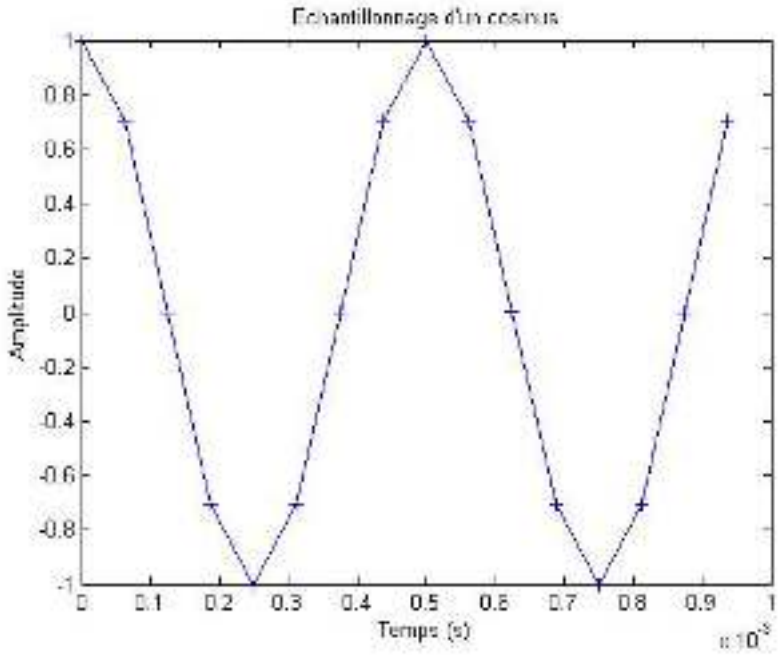
1. Génération d'un signal numérique

Engendrer un signal numérique représentant N échantillons d'une cosinusoïde de fréquence f_0 = 2 kHz, la fréquence d'échantillonnage étant fixée à 16 kHz.

Ce signal échantillonné est en fait une suite de N points tous les $T_e = \frac{1}{f_e}$, que l'on peut donc stocker dans un vecteur de taille N.

```
f0 = 2000; %La fréquence du signal
fe = 16000; %La fréquence d'échantillonage
N = 2*fe/f0; %Le nombre d'échantillons dépend de ces fréquences

t = [0:1:N-1]; %Nous générons un vecteur normé
T = t / fe; %Vecteur de temps [Te  2*Te  3*Te  ...  (N-1)*Te]

Y = cos(2*pi*f0*T); %Y est donc un vecteur de N points représentant un
cosinus échantillonné

plot(T, Y, '-+') %On trace la courbe
title('Echantillonnage d''un cosinus')
xlabel('Temps (s)')
ylabel('Amplitude')

E = (Y*Y')/N %Calcul de l'énergie : Somme des carrés des termes divisée par
le nombre d'échantillons
```

Tracer ce signal en faisant apparaître en abscisse le temps.
Voici le résultat :



On retrouve un comportement sinusoïdal.

Déterminer l'énergie de ce signal
Pour le calcul de l'énergie, MATLAB trouve 0,5J ce qui est la valeur trouvée avec le calcul théorique.

Nous avons essayé de faire varier les fréquences et nous trouvons toujours 0,5J ce qui est normal car si l'échantillonnage est bon, le signal peut être retrouvé et donc il est toujours le même.
Rq : Pour faire le calcul des sommes de carrés, nous utilisons la propriété du produit scalaire d'un vecteur et de sa transposée : $Y.Y' = \sum_k y(k)^2$

2. Transformation de Fourier discrète

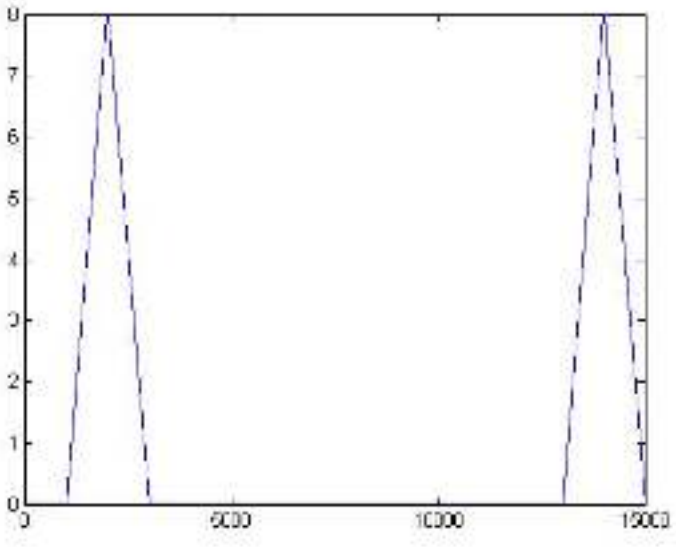
Calculer la transformée de Fourier discrète (TFD) du signal généré en utilisant la formule théorique. Noter les difficultés à considérer pour réaliser l'algorithme.
La formule théorique de la TFD est :

$$X(k) = \sum_{n=0}^{N-1} x(n).e^{-2\pi.\frac{kn}{N}}$$

Nous l'implémentons sous MATLAB grâce à deux boucles for imbriquées :

```
TFD=1:N; %On crée un vecteur de N points
TFD=TFD*0; %On initialise tous les points à 0

W=exp(2*pi*j/N); %Il s'agit d'un terme constant que nous calculons à la
puissance kn
for k=1:N %On calcule X(k)
    Wk=W^(k-1);
    for i=1:N %On fait la somme de 0 à N-1
        TFD(k) =TFD(k) + Y(i)*Wk^(-i+1);
    end
end

figure
plot((0:N-1)*f0/2,abs(TFD))
title('TFD calculée du cosinus')
xlabel('Fréquence (Hz)')
ylabel('Amplitude')
```

Tracer le module du spectre correspondant en fonction de la fréquence. Observer et justifier le spectre obtenu. En particulier, préciser l'incrément fréquentiel et montrer l'influence du nombre de points dans la déduction de la fréquence du signal.
On obtient ce spectre :



On remarque deux pics : l'un à 2000 Hz et l'autre à 14000 Hz. Soit f_0 et $f_e - f_0$. C'est le résultat attendu lors de la transformation de Fourier d'un cosinus échantillonné.

Le spectre d'un cosinus « pur », ne comporte que deux pics : l'un à f_0 et l'autre à $-f_0$. Or l'échantillonnage translate la partie négative du spectre de f_e. D'où le pic à $f_e - f_0$.

Remarquer et justifier l'amplitude des raies ; déterminer l'énergie de ce spectre. Conclure.
L'amplitude des raies est de 8 du fait de la conservation d'énergie. Ainsi l'énergie est : $\frac{8}{N} = 8.\frac{2f_e}{f_0} = 0{,}5J$
La conservation de l'énergie, ainsi que le théorème de Parseval est donc vérifié.

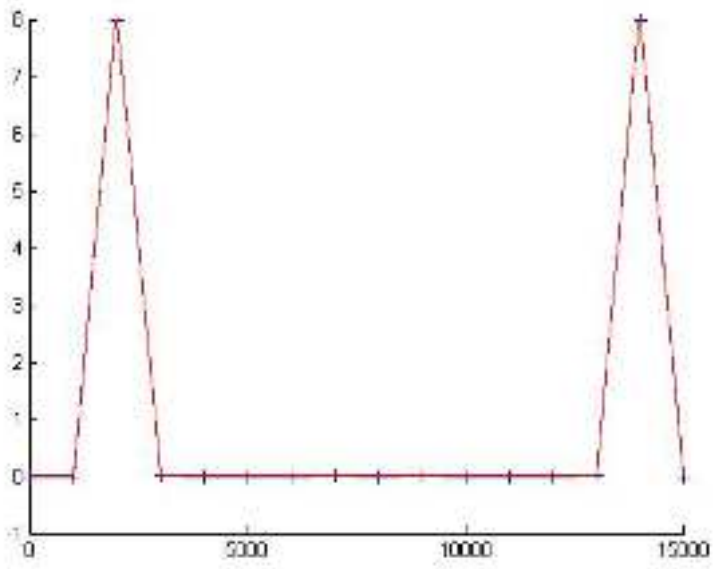
Remarque : La largeur des raies est importante à cause de f_e. En effet, nous n'avons pas assez de points (16 sur un intervalle de 16000Hz) et c'est pour cela que les raies sont larges de 2000Hz. L'incrément fréquentiel est donc : $\Delta f = \frac{f_e}{N} = 1000Hz$

3. Transformée de Fourier Rapide

*Comparer le résultat précédent avec celui obtenu à l'aide de l'algorithme de la Transformée de Fourier Rapide (*fonction fft() de MATLAB*).*

```
Z = fft(Y)
figure
hold on %On va aficher les deux courbes sur la même figure
plot((0:N-1)*f0/2,Z,'-+') %On trace la fft calculée par MATLAB en bleu

plot((0:N-1)*f0/2,abs(TFD),'r') %On trace celle que nous avons calculé plus
haut en bleu
legend('Calcul avec fft()','TFD calculée')
title('Comparaison entre la TFD calculée et le calcul avec fft()')
xlabel('Fréquence (Hz)')
ylabel('Amplitude')
```

Nous obtenons la courbe suivante immédiatement :



On voit que les deux courbes sont presque confondues sauf à l'extérieur des raies où l'on voit que la TFD calculée par MATLAB dévie un peu de 0.

La fonction fft() est donc un peu moins précise que notre algorithme mais par contre elle est plus rapide au calcul.

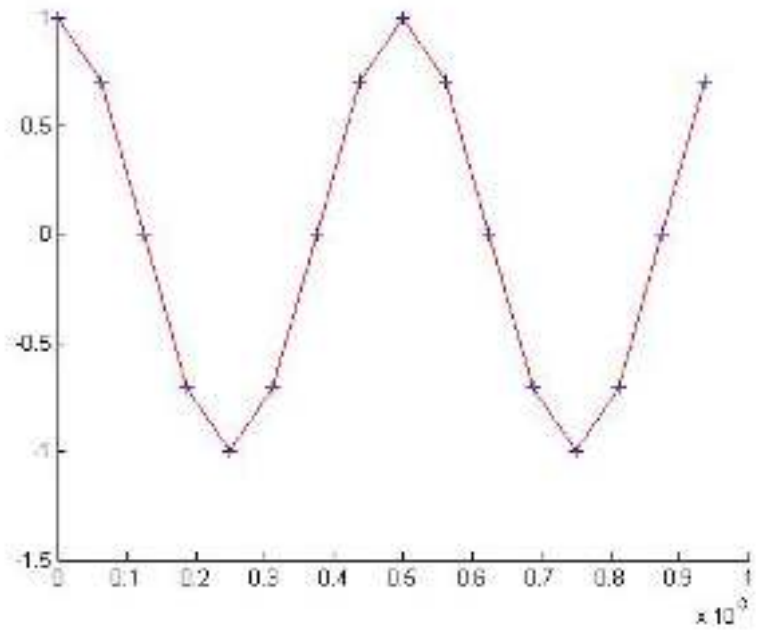
4. Transformée de Fourier Inverse - Reconstruction

Comparer le signal reconstruit à l'aide de la FFT inverse au signal original de la question 1.

Nous écrivons un spectre qui va afficher les deux courbes afin de les comparer :

```
figure
hold on
plot(T, Y, '-+') %On affiche notre signal échantilloné
plot(T, ifft(TFD), 'r') %On affiche la reconstruction
legend('Signal initial','Calcul avec ifft()')
title('Comparaison entre le signal initial et la reconstitution')
xlabel('Temps (s)')
ylabel('Amplitude')
```

On obtient cette courbe :



On remarque encore deux courbes confondues. Les deux signaux sont donc identiques, en phase, en fréquence, comme en amplitude. La transformation de Fourier inverse et la transformation de Fourier sont donc bien des applications réciproques.

5. Théorème de Shannon

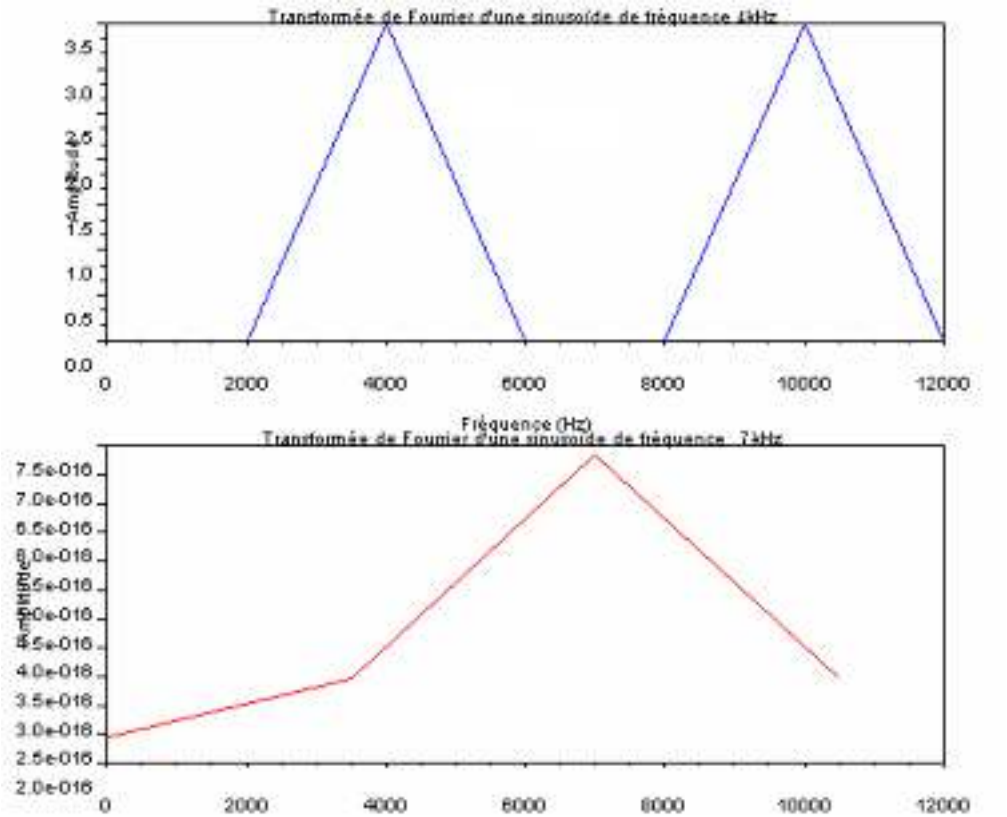
Tracer les spectres des signaux numériques sinusoïdaux de fréquence : 4 kHz et 7 kHz. Justifier les résultats obtenus.

Nous rédigeons un script MATLAB qui va nous permettre de comparer les deux courbes :

```
f1 = 4000;
f2 = 7000;
fe = 12000;
```

```
N1 = 2*fe/f1
N2 = 2*fe/f2

t1 = [0:1:N1-1];
T1 = t1 / fe;

t2 = [0:1:N2-1];
T2 = t2 / fe;


Y1 = sin(2*%pi*f1*T1);
Y2 = sin(2*%pi*f2*T2);

figure
plot((t1)*fe/N1,abs(fft(Y1)))
title('Transformée de Fourrier d''une sinusoïde de fréquence 4kHz')
xlabel('Fréquence (Hz)')
ylabel('Amplitude')

figure
plot((t2)*fe/N2,abs(fft(Y2)), 'r')
title('Transformée de Fourrier d''une sinusoïde de fréquence 7kHz')
xlabel('Fréquence (Hz)')
ylabel('Amplitude')
```

Nous obtenons les courbes suivantes :



On voit bien que le théorème de Shannon n'étant pas respecté dans le deuxième cas (car la fréquence maximale du signal, 7 kHz, est supérieure à la moitié de la fréquence d'échantillonnage, 12kHz), le signal est déformé puisqu'on constate l'apparition de composantes de fréquence différentes que les fréquences d'origine. En fait, il s'agit de deux raies qui sont confondues car l'incrément fréquentiel est trop grand. En effet, il est de 4000 Hz donc l'espace entre deux raies n'est pas distinguable.
Il s'agit d'un repliement de spectre.
Pour éviter ce phénomène, il faut veiller à respecter le théorème de Shannon. Cela revient à augmenter f_e et donc augmenter le nombre de points.

Dans le premier cas, l'incrément fréquentiel est assez faible pour identifier les deux raies à f_0 et $f_e - f_0$.

6. Filtre anti-repliement

Un filtre anti-repliement est un filtre passe bas ayant pour fréquence de coupure la moitié de f_e. Ainsi la fréquence maximale du signal obtenu y est bien inférieure. Il respecte donc le théorème de Shannon et il n'y a pas de repliement de spectre.
Ce filtre ne peut pas être numérique, car pour traiter le signal, il faut d'abord l'échantillonner.
Un filtre anti-repliement est donc obligatoirement analogique.

7. Exercice d'approfondissement

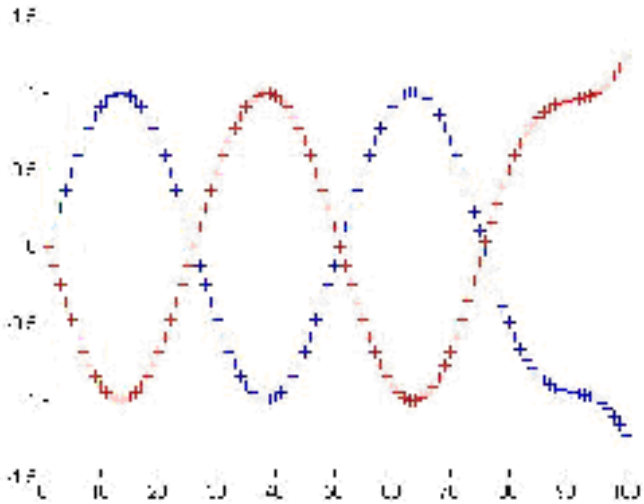
Une sinusoïde de fréquence 200kHz est échantillonnée à la fréquence 1000kHz. Quel est le signal obtenu lors dune reconstruction parfaite ?
Le signal obtenu est le signal original. L'échantillonnage respecte le théorème de Shannon, donc aucune information n'est perdue.

La fréquence d'échantillonnage est maintenant de 250kHz.
On a changé la fréquence d'échantillonnage et le théorème de Shannon n'est plus vérifié. Le signal est donc déformé et les informations sont perdues.

Nous allons maintenant afficher les signaux reconstitués dans les deux cas évoqués précédemment.
Le programme est le suivant :

```
f0 = 200*10^3;

fe1 = 1000*10^3;
fe2 = 250*10^3

N1 = 10
N2 = 10

t1 = [0:1:N1-1];
T1 = t1 / fe1;

t2 = [0:1:N2-1];
T2 = t2 / fe2;


Y1 = sin(2*pi*f0*T1);
Y2 = sin(2*pi*f0*T2);

hold on
plot(interp(Y1,10),'+')
plot(interp(Y2,10),'r+')
```

Nous interpolons (reconstruisons) le signal à partir de 10 échantillons pris du signal original avec deux fréquences d'échantillonnage différentes.



On constate un déphasage de π entre les deux courbes échantillonnées à 1000kHz et 250kHz. La première interpolation apparaît sur la courbe en bleu et la seconde en rouge. On voit bien que le signal qui a été bien reconstruit est celui qui a été échantillonné à 1000 kHz.

8. Pour aller plus loin…

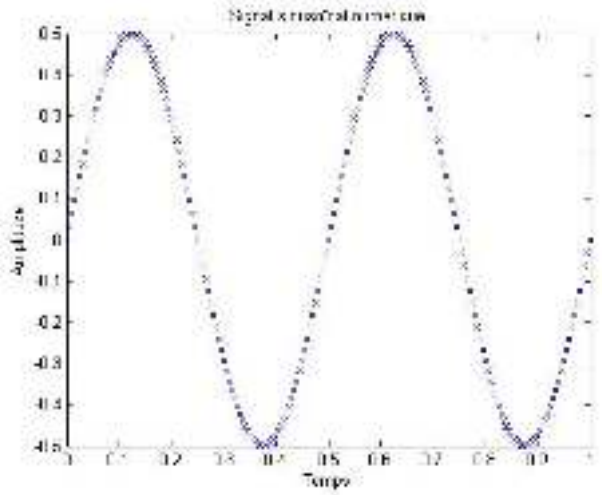
8.1. Générateur sinusoïdal numérique

Un générateur sinusoïdal numérique SIN1 où les paramètres fréquence, durée, amplitude sont réglables

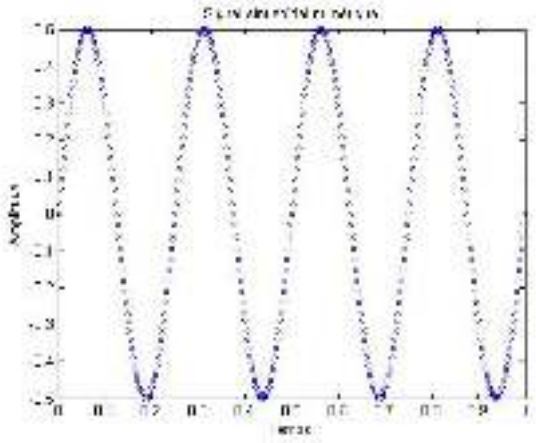
Nous voulons générer une sinusoïde d'une fréquence donnée, d'une durée donnée et d'une amplitude donnée. Pour cela nous allons créer 3 variables qui contiendront ces paramètres.

```
F=2; % Fréquence du signal (Hz)
D=1; % Durée du signal (s)
A=0.5; % Amplitude du signal
N=100; % Nombre de points de l'échantillonage (doit être supérieur à
2 pour respecter le théorème de Shannon)

Fe=N*F; % La fréquence d'échantillonage
T=[0:1/Fe:D] % Le vecteur temps

SIN1=A*sin(2*pi*F*T) % Le vecteur du signal
plot(T,SIN1,'x') % On trace le signal pour vérifier le résultat
```

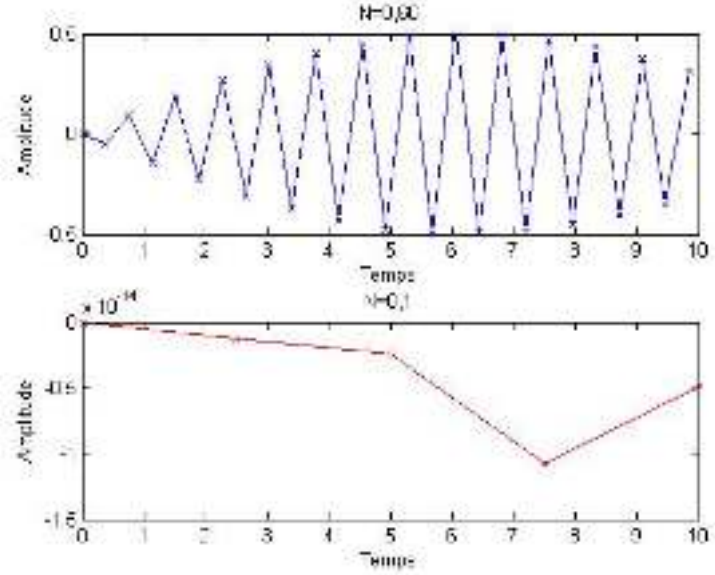
On obtient pour l'exemple donné, un signal de fréquence 2Hz, d'une durée d'1 seconde et d'amplitude 0,5, la courbe suivante :



Et voici le même signal dont on a doublé la fréquence, par exemple :



Il est possible de générer tous les signaux sinusoïdaux avec ce générateur si on respecte toujours la condition de Shannon, N ≥ 2.
Sinon, cela peut faire des signaux qui n'ont plus aucun rapport, par exemple pour N=0,66 et N=0,1 :



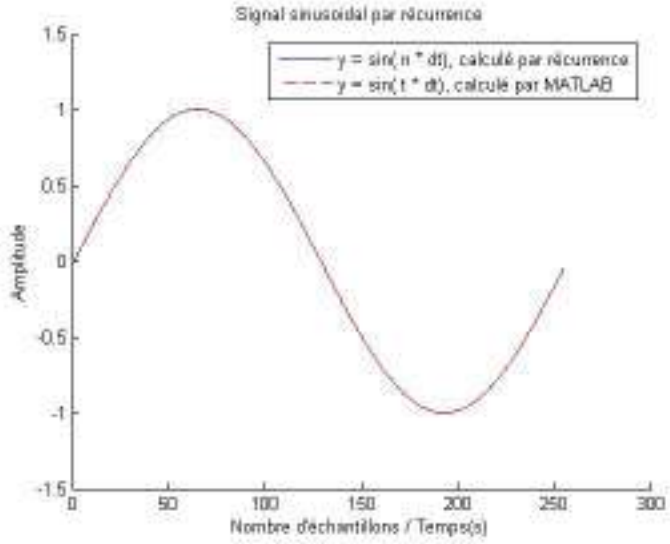
Un générateur sinusoïdal numérique SIN2 de 256 échantillons

Nous allons maintenant utiliser la version récursive du calcul d'un sinus.

```
%Nombre échantillons
N = 256;

%une periode
T = 2*pi;
%intervalle entre l echantillonnage
dt = T / N;

%définition du vecteur temps
t1 = [1:1:N-1];

%valeurs initiales
x(1) = 0;
x(2) = sin(dt);
a1 = 2 * cos(dt);

for n = 3:1:N-1
    x(n) = a1 * x(n-1) - x(n-2);
end

figure
hold on
plot(t1,x,'b')
plot(t1,sin((t1-1)*dt),'r--')
legend('y = sin( n * dt), calculé par récurrence','y = sin( t * dt),
calculé par MATLAB')
title('Signal sinusoidal par récurrence')
xlabel('Nombre d''échantillons / Temps(s)')
ylabel('Amplitude')
```

Nous obtenons la courbe suivante : (voir ci-dessous)

On remarque que les deux courbes sont parfaitement analogues et donc que la formule par récurrence du sinus est valable.



Par contre, il y a des limites à l'utilisation de cette méthode :

- Lorsque l'on veut générer un signal de fréquence élevée sur un certain temps, il se peut que le nombre d'échantillons (256 par période) soit très élevé et donc qu'il faille un temps de calcul important, et un espace de stockage important aussi.

- De plus, si une erreur est faite sur les valeurs initiales, elles sont propagées (voire amplifiées) sur la suite des calculs. Une telle erreur est par ailleurs inévitable dans des calculs en machine comme ceux-ci.

Un générateur sinusoïdal SIN3 de 4096 échantillons comportant les fréquences 193,7988 Hz et 196,4905 Hz.

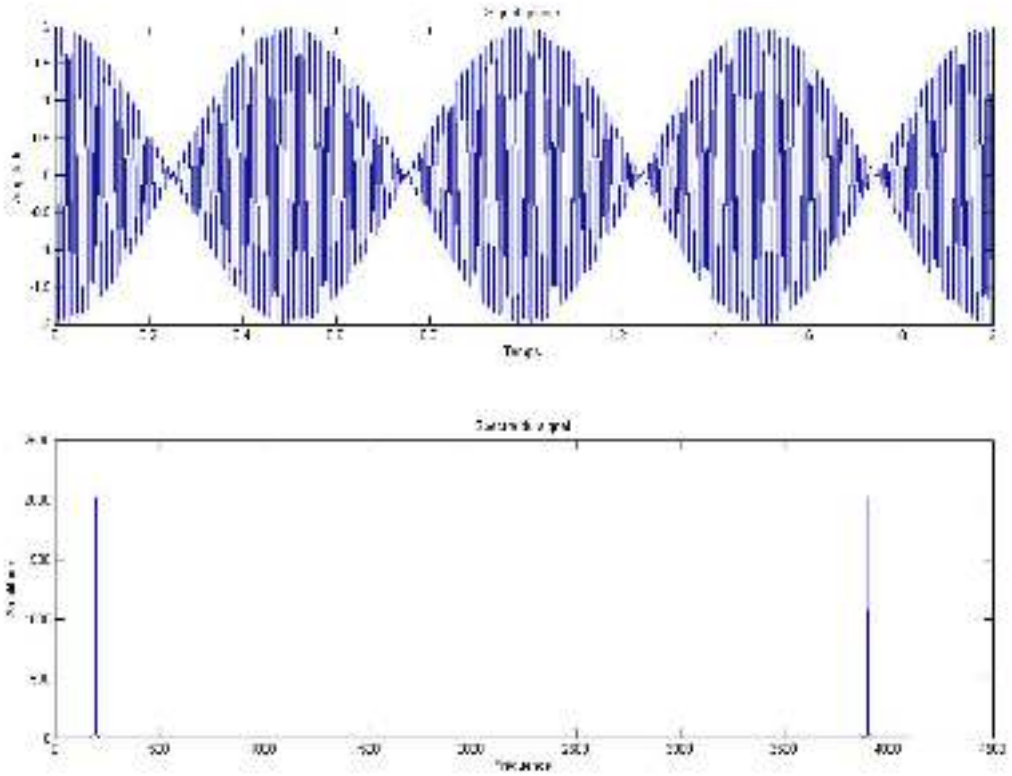
Comme nous n'avons pas d'autres indications, nous considérons que le signal doit être généré sur une seconde.

Ainsi le script est :

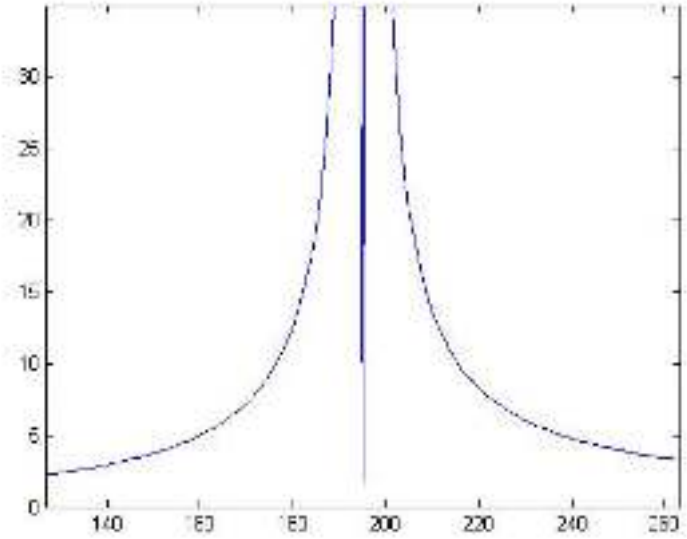
```
clc
close all
clear all

F1 = 192,7988;
F2 = 196,4905;

N=4096;
Fe = N;
t=[0:1/Fe:1];

y = sin(2*pi*F1*t)+sin(2*pi*F2*t);
subplot(211),plot(2*t,y),title('Signal généré'), xlabel('Temps'),
ylabel('Amplitude')
subplot(212),plot(abs(fft(y))),title('Spectre du signal'),
xlabel('Fréquence'), ylabel('Amplitude')
```

Pour que le signal comporte les deux fréquences, on le construit par somme des deux sinusoïdes pures.

Affichons la courbe du signal et son spectre :



Pour mieux observer les fréquences du signal on fait un zoom sur le premier pic…



… qui est en fait double ! En effet, il y a une composante en 196,4905Hz et une composante en 193,7988Hz. C'est normal car notre signal est composé de deux sinusoïdes à ces fréquences.
Le deuxième double pic est situé aux alentours de 3800 car il s'agit de la partie négative du spectre translatée par la fréquence d'échantillonnage (4096Hz).

Ces sinusoïdes peuvent avoir pour fréquence maximale 2048Hz sinon, le théorème de Shannon ne sera pas respecté et le signal ainsi échantillonné sera déformé et on ne pourra pas en déterminer les fréquences qu'il contient.

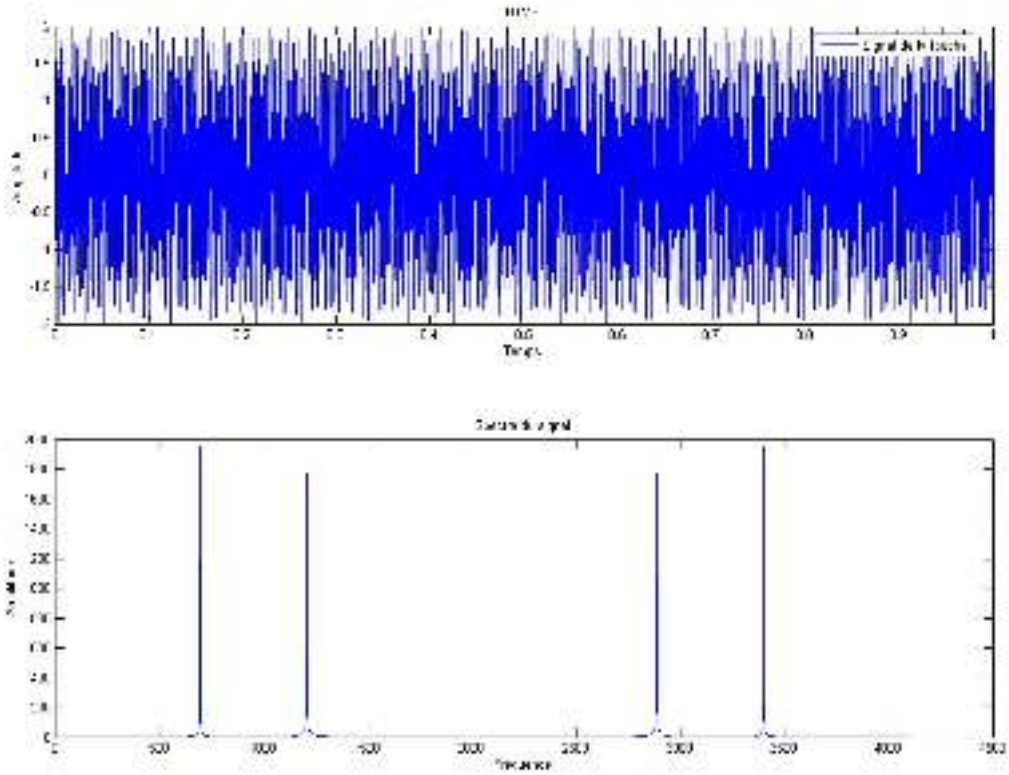
8.2. Signaux DTMF (Dual Tone Multiply Frequency)

Nous allons réaliser un programme qui génère un signal DTM1 correspondant à l'une des touches.

```
touche = input(' \n Saisir une touche de votre clavier \n','s')
if(isempty(touche))
    disp('Erreur vous avez rien saisi')
end

%Fréquences hautes et Basses
hf = [1209 1336 1477 1633];
bf = [697 770 852 941];

%On cherche la ligne :
if(touche == '1' | '2' | '3' | 'A')
  ligne = 1;
end
if(touche == '4' | '5' | '6' | 'B')
  ligne = 2;
end
if(touche == '7' | '8' | '9' | 'C')
  ligne = 3;
end
if(touche == '*' | '0' | '#' | 'D')
  ligne = 4;
end
```

```
%On cherche la colonne :
if(touche == '1' | '4' | '7' | '*')
  colonne = 1;
end
if(touche == '2' | '5' | '8' | '0')
  colonne = 2;
end
if(touche == '3' | '6' | '9' | '#')
  colonne = 3;
end
if(touche == 'A' | 'B' | 'C' | 'D')
  colonne = 4;
end

%Définition du vecteur temps
Fe = 4096; %Pour un échantillonnage valable
t = [0:1/Fe:1];

x = sin(2*pi*(hf(colonne))*t) + sin(2*pi*(bf(ligne))*t);

%On trace :
figure;
subplot(211),plot(t,x,'b'),title('    DTMF    '),    xlabel('Temps'),
ylabel('Amplitude'), legend('Signal de la touche');
subplot(212),plot(abs(fft(x))),title('Spectre      du      signal'),
xlabel('Frequence'), ylabel('Amplitude')
```

On obtient ce signal, pour l'appui sur la touche 1 :

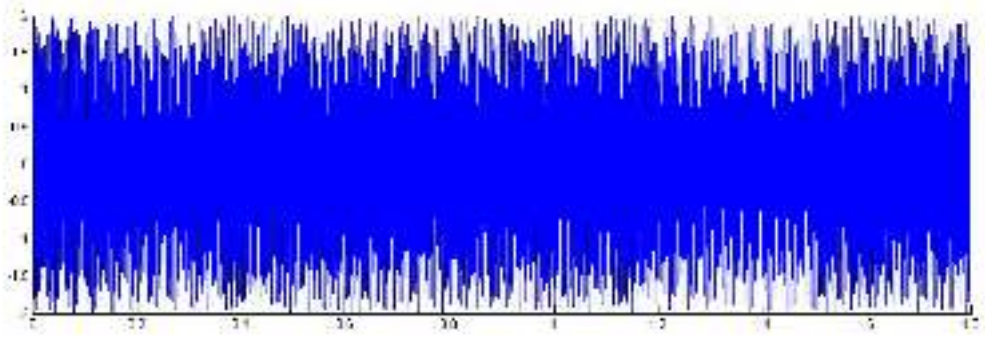


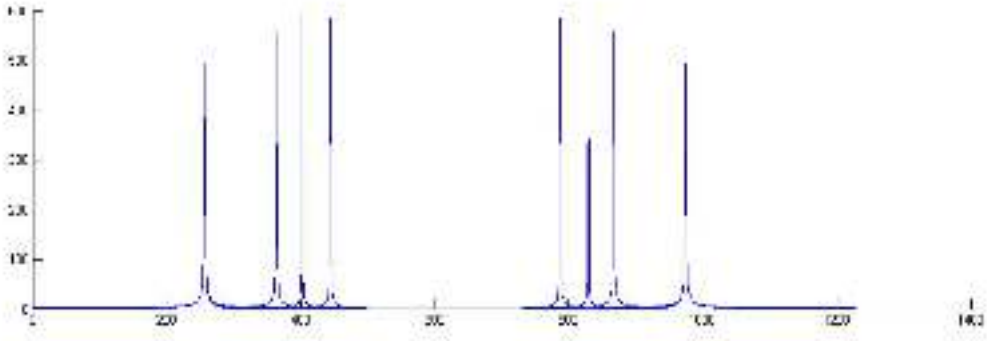
La figure la plus intéressante est le spectre. On voit nettement 4 pics, correspondant à la partie positive du spectre (à gauche, composée d'un pic à 1209Hz et d'un autre à 697Hz) et à la partie négative translatée de la fréquence d'échantillonnage, 4096Hz (à droite, composée d'un pic à 4096 – 1209 = 2887Hz et d'un autre à 4096 – 697 = 3399Hz).

On veut maintenant générer le signal correspondant à l'appui sur les touches « 7 8 9 8 7 9' pendant chacune 300 ms.

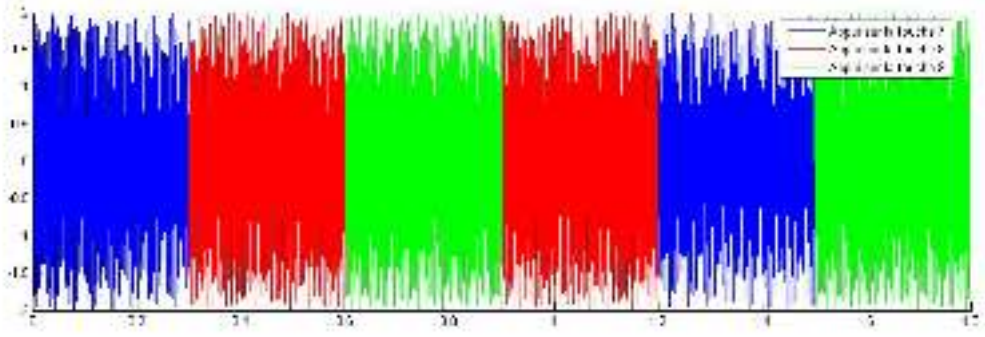
```
clc
clear all
close all

%parametre de l'énoncé
hf = [1209 1336 1477 1633];
bf = [697 770 852 941];

%Séquence de chiffres " 7 8 9 8 7 9"
d=0.3;

%Fréquence échantillonnage
fe = 4096;

%Définition du vecteur temps
n1 = [0:1/fe:d];
n2 = [d:1/fe:2*d];
n3 = [2*d:1/fe:3*d];
n4 = [3*d:1/fe:4*d];
n5 = [4*d:1/fe:5*d];
n6 = [5*d:1/fe:6*d];

%Signaux correspondant à la séquence de touches " 7 8 9 8 7 9"
y1 = sin(2*pi*hf(1)*n1) + sin(2*pi*bf(3)*n1) ;
y2 = sin(2*pi*hf(2)*n2) + sin(2*pi*bf(3)*n2) ;
y3 = sin(2*pi*hf(3)*n3) + sin(2*pi*bf(3)*n3) ;
y4 = sin(2*pi*hf(2)*n4) + sin(2*pi*bf(3)*n4) ;
y5 = sin(2*pi*hf(1)*n5) + sin(2*pi*bf(3)*n5) ;
y6 = sin(2*pi*hf(3)*n6) + sin(2*pi*bf(3)*n6) ;
```

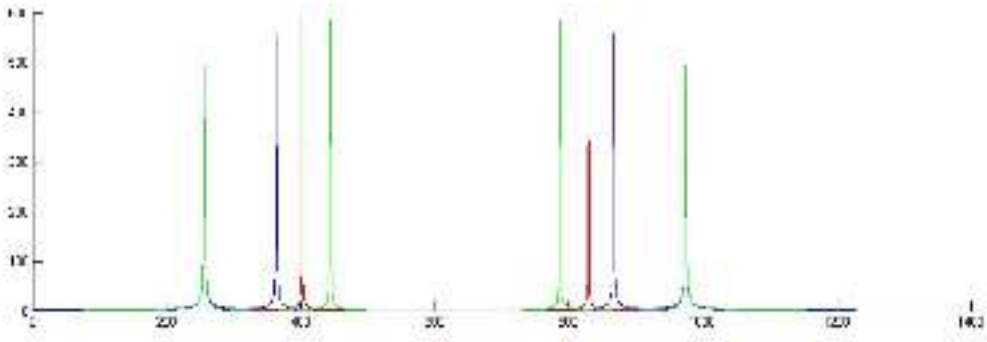
Voici le résultat, la courbe et le spectre associé :





On peut aussi identifier l'appui de chacune des touches pour mieux observer le phénomène :





On voit bien ici la succession de signaux de 300ms. On voit aussi bien apparaître les raies sur le spectre correspondant à chacun des signaux.

Conclusion

Ce TP fut l'occasion de se familiariser avec le logiciel MATLAB au traitement du signal via des exemples simples : représentation de signaux en temps et en fréquence, application des théorèmes de base du traitement numérique du signal (échantillonnage de Shannon), et exemples concrets (signaux DTMF et générateur sinusoïdaux).

Nous avons pu ainsi approfondir le phénomène d'échantillonnage et les problèmes qui y sont associés, comme le repliement de spectre, par exemple.

Ce fut de plus l'occasion de développer une vraie application, le codage DTMF, mettant en pratique autant nos connaissances du logiciel MATLAB que la théorie du traitement du signal. Ce fut l'occasion d'appliquer concrètement des notions théoriques et de se rendre compte de l'importance du traitement du signal dans les transmissions de données puisque c'est l'objet de notre étude, dans notre filière.