Tutoriel transformée de Fourier discrète avec Xcas

1. Introduction

Ce document présente une initiation à la transformée de Fourier discrète et propose d'en explorer quelques aspects de manière concrète au moyen du logiciel de calcul formel **Xcas**[1], **avec une version 1.1.3 ou ultérieure**.

La transformée de Fourier discrète est un outil puissant d'analyse, et au besoin de retouche, des propriétés spectrales d'une fonction f de variable la réelle t. Pour le comprendre, il est plus simple de raisonner d'abord sur une fonction f périodique, de période T. Dans ce cas, les étudiants voient dans les autres disciplines que f présente une signature spectrale faite de raies aux fréquences discrètes $0, \frac{1}{T}, \frac{2}{T}, \frac{3}{T}, \dots -\frac{1}{T}, -\frac{2}{T}, -\frac{3}{T}, \dots$. Ces raies ont pour amplitudes complexes les coefficients de Fourier dits exponentiels :

$$c_k = \frac{1}{T}\int_0^T f(t)e^{-\mathrm{i}2\pi k\frac{t}{T}}dt$$

où $k \in \mathbb{Z}$. En pratique, on calcule rarement ces intégrales. On les approche par la formule des rectangles. On découpe uniformément l'intervalle $[0;T]$ en n petits intervalles de taille $T_e = \frac{T}{n}$, sur lesquels on remplace la quantité à intégrer par sa valeur en l'extrémité gauche. Si bien que :

$$c_k \approx \frac{T_e}{T}\sum_{l=0}^{n-1} f(lT_e)\, e^{-\mathrm{i}2\pi k\frac{lT_e}{T}}$$

ou encore :

$$c_k \approx \frac{1}{n}\sum_{l=0}^{n-1} f(lT_e)\, \omega^{-k.l} \text{ avec } \omega = e^{\mathrm{i}\frac{2\pi}{n}}$$

Autrement dit, et au facteur multiplicatif $\frac{1}{n}$ près, c_k est à peu près le k-ième coefficient de la transformée de Fourier dicrète (TFD) $(F_0, F_1, \dots, F_{n-1})$ de la séquence $(f_0, f_1, \dots, f_{n-1}) = \big(f(0.T_e), f(1.T_e), \dots, f\big((n-1).T_e\big)\big)$. La précision de la méthode est d'autant plus grande que l'exponentielle varie lentement sur la durée T_e, donc que la période $\frac{T}{k}$ de cette exponentielle est grande devant T_e, donc que $\frac{n}{k}$ est grand.

En bref, les premières valeurs $F_0, F_1, F_2, \dots$ de la TFD approchent convenablement les premiers coefficients de Fourier d'indices positifs $c_0, c_1, c_2, \dots$. Les dernières valeurs de la TFD sont tout aussi précieuses : comme $\omega^{-k.l} = \omega^{-(n-k).l}$, les valeurs $F_{n-1}, F_{n-2}, F_{n-3}, \dots$ approchent $c_{-1}, c_{-2}, c_{-3}, \dots$. Plus n est grand, plus l'observation du spectre est large.

Quand la fonction f n'est pas périodique, son spectre est un « continuum » de raies. La TFD approche celles aux fréquences $0, \frac{1}{T}, \frac{2}{T}, \frac{3}{T}, \dots -\frac{1}{T}, -\frac{2}{T}, -\frac{3}{T}$ où T ne désigne plus une période, mais un temps d'observation.

[1] Voir ici : http://www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html

2. Échantillonnage

Il est usuel de nommer « signal » une fonction de variable réelle, cette variable étant assimilée au temps (mesuré en secondes), la valeur de la fonction à l'instant t désignant la grandeur physique à laquelle on s'intéresse (elle peut être de diverses natures : tension, intensité, pression, etc.).

Les signaux traités peuvent être très réguliers ou au contraire assez chaotiques ou « bruités », et ils peuvent être périodiques ou non. Il est cependant commode de considérer tout signal comme périodique, quitte à tronquer le signal sur une durée limitée[2] et à considérer que le signal se répète au-delà de cette durée.

Dans les systèmes numériques, au contraire des systèmes analogiques, le signal n'est pas mesuré en continu mais seulement à intervalles réguliers : c'est ce qu'on appelle l'échantillonnage du signal. L'intervalle de temps entre deux prises de mesure successives s'appelle le **pas d'échantillonnage**. Après calibrage, les mesures sont ensuite disposées sur une échelle discrète et converties en nombres entiers. Cette opération, dite de **quantification,** n'est pas prise en compte dans ce présent document mais elle ferait partie intégrante d'un processus de numérisation d'un signal.

Ici, l'échantillonnage d'un signal sur une certaine durée produira donc une liste de nombres réels. Cette liste peut se manipuler comme un vecteur (ou tableau à une seule ligne).

Quelques commandes Xcas utiles pour créer et manipuler des vecteurs :

Attention, dans Xcas les vecteurs sont indexés à partir du rang 0 (et non 1) !
On peut créer un vecteur dans Xcas en donnant la liste de ses composantes ; ainsi :
`t:=[5,3,8,17]`
crée un vecteur à quatre composantes, la première étant t[0], valant 5 et la dernière étant `t[3]` , valant 17 (`t[4]` n'existe pas, une erreur est signalée).
Il est également possible de créer automatiquement un vecteur (ou liste) ayant des composantes régulièrement espacées, c'est idéal pour représenter la liste des instants de prise de mesure :
seq(0.1 , 3, 0.5)
permet de créer une liste de valeurs démarrant avec la valeur 0,1, avançant de 0,5 en 0,5 et ne dépassant pas 3 (la dernière est donc ici égale à 2,6), on obtient : [0.1, 0.6, 1.1, 1.6, 2.1, 2.6]
On peut enfin créer des vecteurs avec la fonction seq, comme sur les calculatrices. f désignant une fonction,
`t1:=seq(f(k),k,1,15,0.5) ;`
crée la liste des images f(k) pour k variant de 1 à 15 par pas de 0,5
Une fois que des vecteurs sont créés, on peut aisément leur appliquer des calculs. Par exemple :
`v:=0.002*t ;`
donne un vecteur `v` dont les composantes sont égales à celles de `t` multipliées par 0,002[3]. Dans la même logique, l'addition des vecteurs fonctionne :
`w:=v+t`
donne un vecteur dont les composantes résultent de l'addition de celles de `v` et de celles de `t` de même rang (mathématiquement parlant, on obtiendra ici $w = 1{,}002t$).
On peut aussi extraire un sous-vecteur d'un vecteur en spécifiant les indices de début et de fin:
```
w[1..4]
    [0.6012, 1.1022, 1.6032, 2.1042]
```
Pour la multiplication, il faut distinguer le produit scalaire, qui se forme ainsi : `w*t`(pour des vecteurs en ligne), du produit composante par composante, qui s'écrit : `w.*t` , et redonne un vecteur.
On peut enfin « appliquer » une fonction à un vecteur. Par exemple, la fonction carré :
carre_vecteur(x):=x.*x ; définit la fonction d'élévation au carré des composantes d'un vecteur, le nom choisi ici pour

[2] Mathématiquement parlant, c'est une restriction (par exemple à un intervalle d'amplitude $N \times Te$).
[3] C'est l'opération de multiplication d'un vecteur par un scalaire dans un espace vectoriel.

cette fonction est carre_vecteur (ne pas l'appeler « carre », ce mot est utilisé par Xcas en géométrie).
carre_vecteur(t) donne :
`[0.01,0.36,1.21,2.56,4.41,6.76]`
Représentation d'un signal échantillonné avec t désignant la liste des valeurs du temps et s celle des valeurs du signal : pour obtenir un nuage de points, on utilise `scatterplot(t,s)` (ou en français : `nuage_points(t,s)`) et `polygonplot(t,s)` pour obtenir une ligne brisée joignant les points).
Si les données sont exactes, Xcas effectue autant que possible des calculs exacts. Dans le cas d'étude des signaux, pour traiter efficacement les calculs (c'est-à-dire rapidement et sans utilisation de trop de mémoire), on fera du calcul approché, que l'on forcera en utilisant la notation scientifique (par exemple : 2e-3 au lieu de 2*10^-3) ou la notation décimale, même pour les entiers, par exemple 2.0 au lieu de 2.

Exemples de création et d'échantillonnage d'un signal

Une bonne base pour créer des signaux est fournie par les fonctions sinusoïdales (une alternative intéressante étant fournie par les fonctions en « créneaux » ou en « dents de scie »).

La période (ou pas) d'échantillonnage est notée T_e. On en déduit la fréquence d'échantillonnage, $F_e = \frac{1}{T_e}$, qui se mesure en Hz, ainsi que la pulsation d'échantillonnage $\omega_e = 2.\pi.F_e$, qui se mesure en rad.s^{-1}.

On commence à mesurer le signal à l'instant $Nmin \times T_e$ et on termine à l'instant $Nmax \times T_e$, le nombre de mesures étant ainsi $Nmax - Nmin + 1$ et le nombre d'intervalles $Nmax - Nmin$.

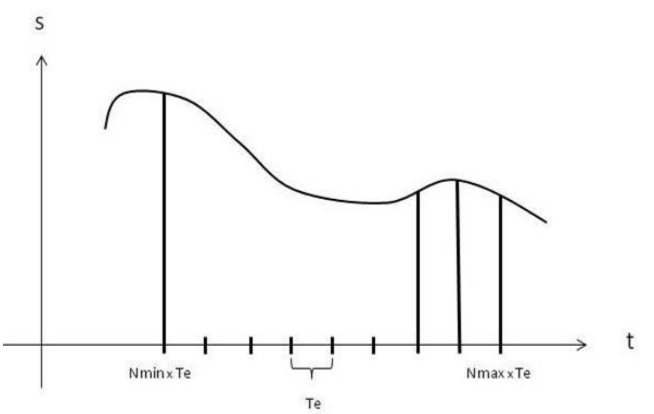


Figure 1 : Graphique pour l'échantillonnage temporel (amplitude s *du signal en fonction du temps)*

Il est très intéressant de travailler sur des données sonores, parce que Xcas permet de les traiter commodément. Pour s'en tenir à un standard, le son peut être échantillonné à 22050 Hz[4]. On peut proposer comme **activités :**

1. **créer un son d'une demi-seconde simulant une note LA 440 (fréquence 440 Hz) de différentes « couleurs » (son plus ou moins pur), et en représenter graphiquement le premier centième de seconde (comme courbe ou comme série de points isolés) ;**
2. **charger un son à partir d'un fichier, et le tronquer ou en extraire une section d'un quart de seconde ;**
3. **charger un son et transformer sa « hauteur » en doublant la fréquence de base.**

Exercice 1. On peut créer le vecteur des instants d'échantillonnage en employant la fonction `soundsec` :

```
t :=soundsec⁵(0.5,22050);
```

Il revient exactement au même de définir la fréquence et le pas d'échantillonnage puis le vecteur :

```
Fe:=22050;  Te:=1/Fe;  t:=seq(k,0.5-Te,Te);
```

Par exemple pour une durée d'une seconde avec une fréquence d'échantillonnage de 10 Hz

soundsec(1.5 , 10) donne [0.0 , 0.1 , 0.2 , 0.3 , 0.4 , 0.5 , 0.6 , 0.7 ,0.8 , 0.9 , 1.0 , 1.1 , 1.2 , 1.3 , 1.4].

Nous pouvons maintenant créer le signal sinusoïdal (son pur) de fréquence 440 Hz, échantillonné à 22050 Hz :

```
s:=sin(440.0*2*pi*t);⁶
```

[4] Un son de bonne qualité (CD) devait plutôt être échantillonné à 44100 Hz. Un son de qualité téléphonique peut être échantillonné à 7000 Hz.
[5] Commande disponible à partir de la version de novembre 2014 du logiciel Xcas.
[6] Travailler avec 440.0 au lieu de 440 « force » les calculs numériques.

Un centième de seconde, à 22050 Hz, représente 220 points d'échantillonnage, figure 2 :

```
scatterplot(t[0..220],s[0..220]);
```

Penser à cliquer sur le bouton « Auto/ Autoscale(full) » pour redimensionner la fenêtre. On obtient la « courbe » ci-dessous (ce n'est pas - en toute rigueur - la « courbe » de la fonction sinus d'origine, mais le nuage des points résultant de l'échantillonnage). Pour faire abstraction du temps, utiliser l'instruction qui suit, figure 3.

```
scatterplot(s[0..220])
```

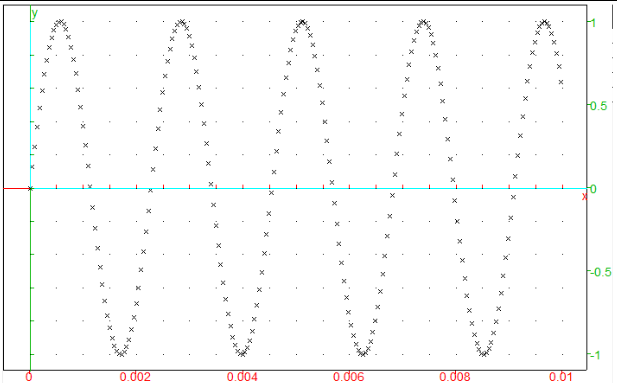


Figure 2 : Un centième de seconde d'un son sinusoïdal à 440 Hz

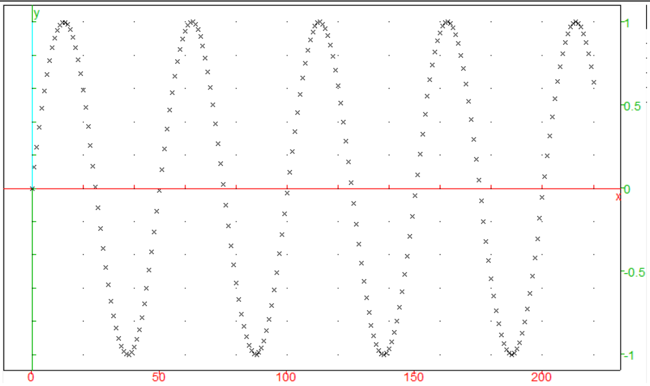


Figure 3 : Encore un centième de seconde, mais en abscisse, le numéro de la donnée

On verra dans l'exercice 2 (à l'écoute) que ce son n'est pas très mélodieux : c'est un son de diapason, dont le timbre est pauvre. Les musiciens préfèrent souvent des sons basés sur une fonction en « dents de scie », qu'il est facile de créer à partir du sinus :

```
s1:=asin(sin(440*2*pi*t));
```

```
scatterplot(s1[0..220]);
```

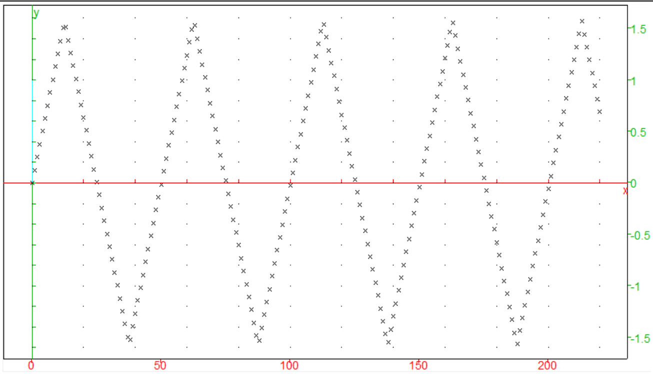


Figure 4 : Un signal en « dents de scie »

(la fonction arcsinus, employée ici, tend à « redresser » le sinus puisqu'elle est réciproque de celle-ci entre $-\frac{\pi}{2}$ et $\frac{\pi}{2}$).

On peut même envisager des sons basés sur une fonction créneau, dont un rapport cyclique modulera la largeur des impulsions.

```
alpha := 0.35;
s2:= sign(s+alpha);
```

```
polygonplot(s2[0..220]);
```

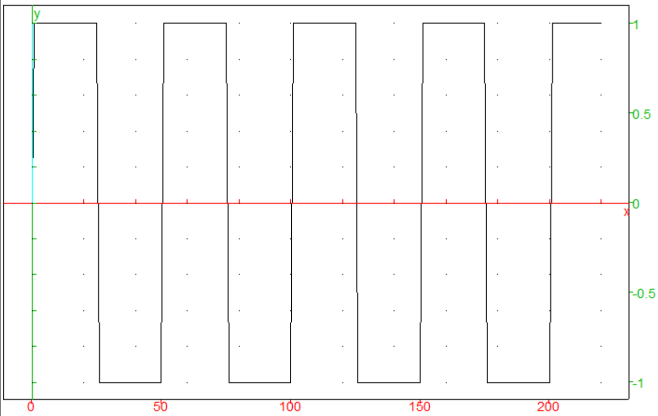


Figure 5 : Un signal « rectangulaire »

Exercice 2. On récupère le fichier son avec la fonction readwav d'Xcas. <u>Attention, le fichier à lire, ici le fichier « chimes.wav », doit être placé dans le même dossier du disque dur que la session en cours, celle-ci doit donc être enregistrée (ou « sauvée ») avant d'exécuter les instructions :</u>

```
y:=readwav("chimes.wav");
```

La variable y qui en résulte est une liste de listes : y[0] contient les paramètres d'échantillonnage, ici la liste [2,16,22050,55684], il y a 2 canaux, 16 bits (ou 2 octets) par canal, la fréquence d'échantillonnage est de 22050 et la taille du fichier est de 55684 octets (donc 27842 par canal, soit 0.63 secondes d'enregistrement a raison de 2 octets par échantillon sonore en mono).

y[0,2] est le deuxième élément (en partant de 0) de la liste y[0], il vaut 22050. y[1] est la liste qui contient les données du 1er canal, y[2] celles du 2ème canal.

Puisque le son est enregistré à la fréquence de 22050 Hz, un quart de seconde contient 5512 échantillons ; on peut donc extraire le son du premier canal ainsi :

```
s:=y[1,0..5511];
```

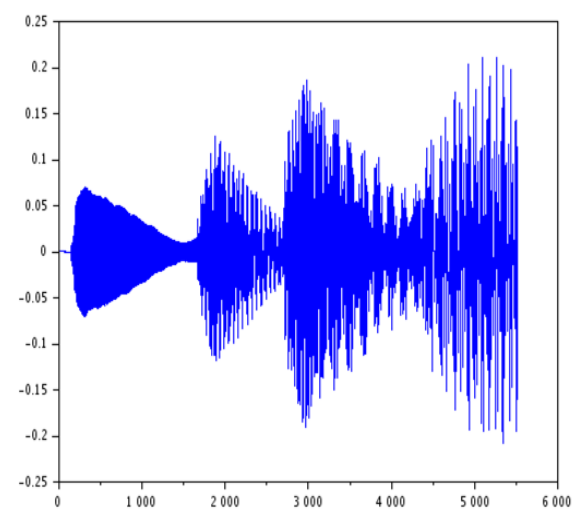


Figure 6 : un signal musical tronqué

Exercice 3. On peut par exemple poursuivre à partir du fichier sonore obtenu dans l'exercice 2. Une manière très rudimentaire - mais inexacte - de « doubler » la hauteur du son consiste à éliminer un échantillon sur deux, en compressant le tout (la sinusoïde se resserre) : la durée est donc moitié moindre[7] :

Pour récupérer dans v1 les termes de rangs pairs d'un vecteur v, on peut définir la fonction :

```
  rang_pair(v)  := seq(v[2k],k,0,floor(size(v)/2));
```

où floor désigne la partie entière, et size donne le nombre d'éléments du vecteur. Dans le cas du signal cela donnera ainsi :

```
y[1]:=rang_pair(y[1]) ; y[2]:=rang_pair(y[2]) ;
```

Pour entendre ce nouveau son, il faudra réduire le nombre de données (deux fois moins), mais conserver la même fréquence d'échantillonnage. Cette première méthode n'est pas illustrée dans les fichiers Xcas joints.

Une autre manière, plus simple et sans perte d'information, consiste à doubler la fréquence d'échantillonnage de restitution du son (par défaut c'était 22050 Hz, on passe à 44100 Hz) :

```
y[0] :=[2,16,44100, 55684] ;
```

ce qui revient au même que de modifier l'élément d'indice 2 de la liste y[0] :

```
y[0,2] :=44100 ;
```

Une fois la transformation faite, on enregistre un nouveau fichier wav avec l'instruction : writewav

```
writewav("chimes_1",y);
```

On écoute le fichier avec un logiciel quelconque de lecture de musique, sa durée sera deux fois moins longue que celle du fichier original. Depuis la version de novembre 2014, on peut aussi lancer l'écoute directement depuis Xcas à l'aide de l'instruction playsnd( ). Par défaut Xcas considère que le son est échantillonné à 44100 Hz.

[7] L'opération visant à « diviser par deux » la fréquence est aussi envisageable, en dupliquant tous les échantillons (au prix d'une piètre qualité).

3. Transformée de Fourier discrète

Les notations utilisées :

Le signal continu initial présente un spectre de fréquences positives occupant une bande limitée (dite monolatérale) comprise entre f_{min} et f_{max}. Le théorème de Nyquist-Shannon impose une fréquence d'échantillonnage F_e (nombre d'échantillons par seconde) au moins égale à $2.f_{max}$.

La période d'échantillonnage est notée T_e (et on a : $F_e = \frac{1}{T_e}$). On choisit le même nombre d'échantillons N pour le signal et pour son spectre[8]. L'échantillonnage de la transformée de Fourier discrète ramène celle-ci à N fréquences distinctes : $0 = 0 \times \frac{F_e}{N}$, $\frac{F_e}{N}$, …, $(N-1) \times \frac{F_e}{N}$. Noter que la résolution spectrale est l'inverse de la durée d'observation : $\frac{1}{NT_e} = \frac{F_e}{N}$.

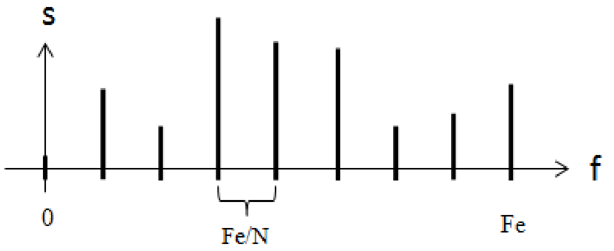


Figure 7 : échantillonnage fréquentiel (amplitude S des fréquences du signal en fonction de la fréquence f)

Quelques commandes Xcas utiles :

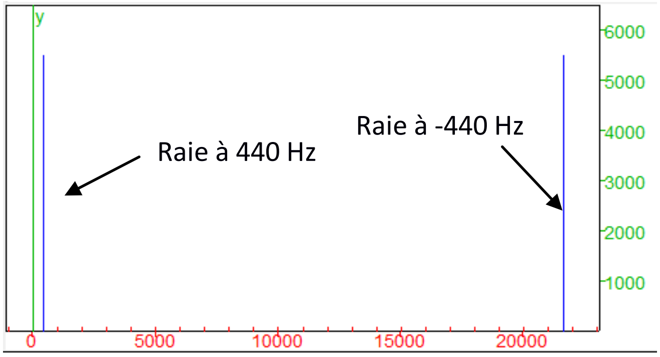
`fft( )` : transformée de Fourier rapide (fast Fourier transform)[9], à appliquer à un vecteur ; retourne un vecteur complexe de même taille (utiliser la commande `abs( )` pour en avoir le module).

`ifft( )` : transformée de Fourier inverse

Exemple : reprenons le son sinusoïdal à 440 Hz d'une durée de 0,5 seconde, employé ci-dessus

```
Fe:=22050; Te:=1/Fe;
t:=seq(0..0.5-Te,Te) ;// vecteur des instants
s:=sin(440*2*pi*t); // vecteur échantillon
N :=size(t); // taille du vecteur
// échantillonnage de la TFD
fr:=seq(0..(N-1),1)*Fe/N;// vecteur des fréquences
// calcul de la TFD
y :=abs(fft(s));
// représentation graphique
batons(fr,y);
```

[8] Sauf à faire du « zéro padding », c'est-à-dire à border la séquence temporelle d'échantillons de valeurs nulles.
[9] La FFT implémente un algorithme de calcul rapide de la TFD, optimal quand N est une puissance de 2.

La fonction *batons* dessine le diagramme en bâtons. Le graphique fait apparaître deux « barres » et (apparemment) rien de plus[10] ; les deux « pics » correspondent aux deux exponentielles qui composent le sinus. La « raie » à 440 Hz est bien visible. Celle à 21610 Hz est en fait l'image périodisée de 22050 Hz (fréquence d'échantillonnage) de celle à -440 Hz (non visible sur l'écran).	 *Figure 8 : Une transformée de Fourier discrète très simple. Pour obtenir cette image, dans le menu cfg de configuration du graphique, imposer comme maximum pour x : WX+=25 000 et comme maximum pour y : WY+=6500.*

Activités.

1. **On propose de créer (ou plutôt simuler) un signal de 0,6 s, échantillonné à 1000 Hz et contenant deux fréquences pures à 40 Hz et 90 Hz d'amplitude 1, déphasées, puis un signal « bruité » déduit du précédent en ajoutant un bruit aléatoire gaussien centré de variance 1, et de comparer les transformées de Fourier discrètes des deux signaux.**
2. **Analyser l'influence de la durée d'observation.**
3. **Proposer une méthode simple de débruitage fondée sur la TFD.**

Nombres aléatoires dans Xcas : la commande `randvector(N, loi_normale)` renvoie un vecteur de N valeurs pseudo-aléatoires suivant une loi normale de moyenne nulle et d'écart-type 1.

Exercice 1. On commence par le signal « déterministe ».

```
//Composantes fréquentielles d'un signal
Fe:=1000.0; Te:=1/Fe;
//observer 0,6 s, échantillonner à Fe
t:=seq(0..0.6-Te,Te);
N:=size(t); //nombre d'échantillons
s:=sin(2*pi*40*t)+sin(2*pi*90*t+pi/4);
//création d'un signal échantillonné.
y:=abs(fft(s));
//compte tenu de la symétrie de y, on ne
garde que N/2 points,
// et on légende l'axe des fréquences avec
les bonnes graduations, voir l'exemple.
N1:=floor(N/2);
fr:=seq(0.. N1-1,1)*Fe/N;
batons(fr[0:N1-1],y[0:N1-1]);
```

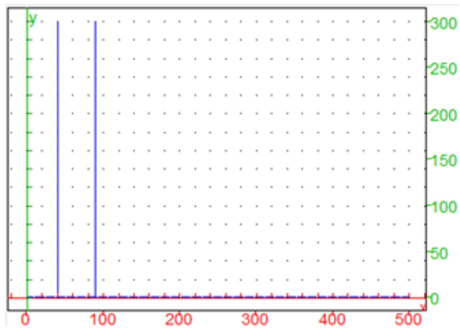


Figure 9 : TFD d'un signal composite à 40 Hz et 90 Hz

Le signal bruité s'obtient par superposition (ou addition).

```
R:=randvector(N,loi_normale);// création
d'un bruit aléatoire centré de variance 1
s_bruit:=s+R;
y_bruit:=abs(fft(s_bruit));
batons(fr,y_bruit[0..N1-1]);
```

On voit nettement se dégager les « raies » à 40 Hz et 90 Hz.

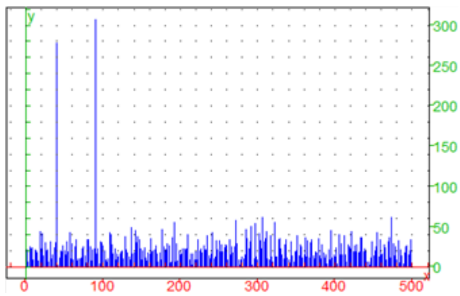


Figure 10 : TFD du même signal, mais bruité avec les « raies » principales à 40 et 90 Hz

[10] En fait, la transformée de Fourier discrète y n'est pas tout à fait nulle en-dehors des deux « pics » parce que la fréquence propre du signal (440 Hz) n'est pas un diviseur exact de la fréquence d'échantillonnage (22050 Hz). Ce point sera discuté plus loin.

Exercice 2. De manière générale, plus la durée d'observation du signal est grande, plus la résolution fréquentielle est fine. Un cas d'école est celui du signal périodique : la TFD renvoie le spectre exact aux fréquences d'échantillonnage (et, donc, les coefficients de Fourier) si le signal est observé sur un multiple entier de sa période, et si la fenêtre d'observation du signal est elle-même un multiple entier du pas d'échantillonnage[11]; sinon le spectre n'est qu'approché. Il gagne néanmoins en netteté à mesure que la fenêtre d'étude s'allonge.

Ici, 0,6s est multiple de la période du signal composite (non bruité) puisque 40x0,6 et 90x0,6 sont des entiers. La durée d'observation est un multiple entier du pas d'échantillonnage puisque 1000x0,6 est entier. La TFD a donc retourné 4 raies pures, correspondant aux fréquences 40 Hz, 90 Hz, 910 Hz[12], 960 Hz[13].

Observons maintenant le signal pendant 0,533s ; 1,11s ; 5,333s, et affichons la TFD correspondant à la plage de fréquences 0 - 150 Hz. Nous constatons les effets annoncés, sous la forme d'une « robe » élargissant les pics.

```
Fe:=1000.0; Te:=1/Fe;
t1:=seq(0, 0.533-Te,Te); t2:=seq(0,1.1-Te,Te); t3:=seq(0, 5.333-Te,Te);
s1:=sin(2*pi*40*t1)+sin(2*pi*90*t1+pi/4);
s2:=sin(2*pi*40*t2)+sin(2*pi*90*t2+pi/4);
s3:=sin(2*pi*40*t3)+sin(2*pi*90*t3+pi/4);
N1:=size(t1); N2:=size(t2);N3:=size(t3);
fr1:=seq(0,(150*N1/Fe),1)*Fe/N1; fr2:=seq(0,(150*N2/Fe),1)*Fe/N2;
fr3:=seq(0,(150*N3/Fe),1)*Fe/N3;
y1:=abs(fft(s1)); y2:=abs(fft(s2)); y3:=abs(fft(s3));
batons(fr1,y1[0..(150*N1/Fe)]);
batons(fr2,y2[0..(150*N2/Fe)]);
batons(fr3,y3 [0..(150*N3/Fe)]);
```

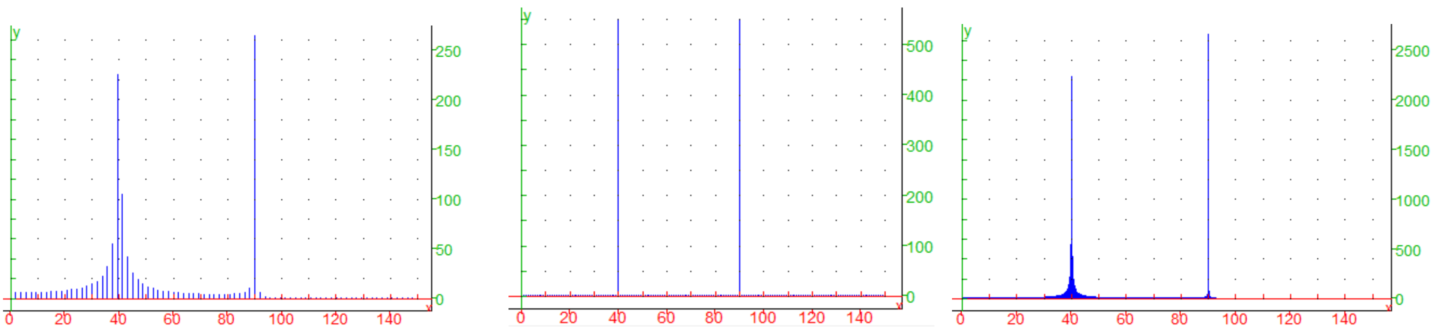


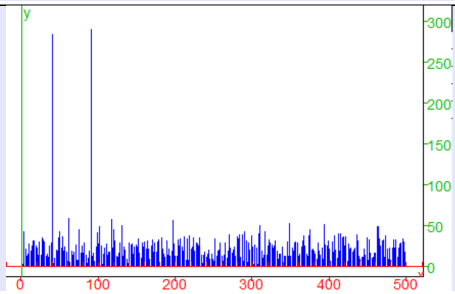
Figure 11 : Influence de la fenêtre d'étude sur la TFD

[11] Cette deuxième condition est, en pratique (c'est-à-dire sur un montage matériel), toujours réalisée
[12] 910 = -90 [1000]
[13] 960 = -40 [1000]

Exercice 3. Travaillons sur la séquence bruitée. Voici ce que l'on obtient en enlevant les fréquences de faible amplitude : au-dessous d'un certain seuil choisi (ici 55), ces fréquences ont une amplitude ramenée à 0 (fenêtre temporelle de durée 0,6s).

```
y1:=fft(s_bruit);
y_debruite:=seq(y1[k]*(abs(y1[k])>55),k,0,N-1)
batons(fr,abs(y_debruite[0..N1-1]));
s_debruite :=ifft(y_debruite);
```

À gauche le spectre du signal bruité, à droite celui du spectre partiellement débruité.



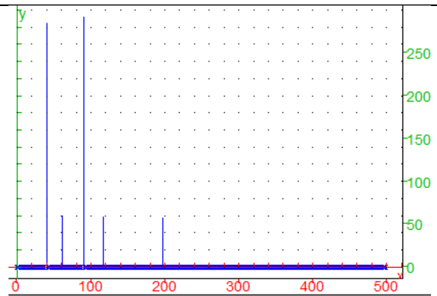


Figure 12 : Débruitage par un procédé rudimentaire

En appliquant la transformée de Fourier inverse, le signal est filtré. Il demeurera toujours une petite erreur sur `s_debruite`. La faute au bruit entachant les deux pics, qui ne peut être supprimé facilement : le bruit a lui aussi une composante spectrale aux fréquences 40 Hz et 90 Hz, qui s'est indissociablement mêlée à celle des sinusoïdes.
Cette méthode est rudimentaire, et peut être source d'erreurs : quand la durée d'observation n'est pas multiple de la période du signal, ou quand le signal est non périodique, toutes les composantes spectrales peuvent être porteuses d'informations utiles.

L'équipe qui a rédigé ce document ressource était composée de :

Cabane Robert, IGEN, groupe des mathématiques
Dubouloz Georges, IA-IPR de mathématiques, académie de Grenoble
Fleurant Sandrine, IA-IPR de mathématiques, académie de Nantes
Zayana Karim, IGEN, groupe des mathématiques
Avec l'aimable participation de Parisse Bernard, de l'université Joseph Fourier de Grenoble.