Chapitre 2

Traitement du signal

2.1 Séries de Fourier

Avant de rentrer dans le vif sujet et d'étudier la transformée de Fourier, nous allons nous arrêter un instant sur les séries de Fourier. Ces dernières permettent de comprendre les concepts qui sous-tendent toute l'analyse de Fourier pour le traitement du signal que nous verrons dans ce cours et sont, selon moi, plus intuitives pour aborder ces questions.

Définition

Pour une fonction $f(x)$ définie sur $[-\pi, \pi]$, on peut écrire sa décomposition en série de Fourier de la façon suivante :

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx) + b_n \sin(nx). \tag{2.1}$$

Nous allons maintenant utiliser Matlab pour obtenir les coefficients a_n et b_n.

Exercice 2.1

1. Ecrire $\int_{-\pi}^{\pi} f(x)\cos(mx)\,\mathrm{d}x$ à l'aide de l'équation (2.1).
2. Décomposer l'expression précédente et isoler les 3 termes :

$$P_1 = \frac{a_0}{2}\int_{-\pi}^{\pi} \cos(mx)\,\mathrm{d}x,$$

$$P_2 = \sum_{n=1}^{\infty} a_n \int_{-\pi}^{\pi} \cos(nx)\cos(mx)\,\mathrm{d}x,$$

$$P_3 = \sum_{n=1}^{\infty} b_n \int_{-\pi}^{\pi} \sin(nx)\cos(mx)\,\mathrm{d}x.$$

3.a Utiliser une représentation graphique sous Matlab pour trouver la valeur de P1.
3.b Utiliser une représentation graphique sous Matlab pour trouver la valeur de P2.
3.c Utiliser une représentation graphique sous Matlab pour trouver la valeur de P3 pour $n \neq m$.
3.d Utiliser une représentation graphique sous Matlab pour trouver la valeur de P3 pour $n = m$.

4. Déduire de ce calcul la valeur des coefficients a_n et monter que $a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,\mathrm{d}x$, pour tout n.

5. Effectuer la même démarche pour obtenir la valeur des coefficients b_n et monter que $b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,\mathrm{d}x$, pour tout n.

Notation exponentielle

Dans l'exercice précédent nous venons de voir que l'on peut décomposer une fonction définie sur $[-\pi, \pi]$ à l'aide de fonctions sinus et cosinus. Il est possible de faire cette décomposition de manière plus compacte à l'aide de la notation exponentielle. On écrira alors $f(x)$ définie sur $[-\pi, \pi]$ sous la forme :

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{in\pi x}. \tag{2.2}$$

Notons que l'on peut étendre à des fonctions $f(x)$ définie sur $[-L, L]$ par :

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{n\pi}{L}x}. \tag{2.3}$$

Exercice 2.2

En utilisant la relation d'Euler : $e^{i\theta} = \cos\theta + i\sin\theta$, démontrer que :

a. $c_n = \frac{a_n - ib_n}{2}$ pour $n > 0$
b. $c_n = \frac{a_n + ib_n}{2}$ pour $n < 0$.
c. $c_0 = \frac{a_0}{2}$.

2.2 Application à la synthèse de signaux sous Matlab

Très bien, nous avons donc maintenant compris que les séries de Fourier correspondent à des décompositions d'un signal sur la *base* des fonctions trigonométriques. Essayons de faire un exemple sous Matlab.

Exercice 2.3 : Effet la somme non-infinie sur les séries de Fourier

1. Créer un script qui va réaliser les opération suivantes :

   - Définir la taille L de l'échantillon utilisé.
   - Créer un vecteur `time` $100L$ points régulièrement espacés entre 0 et L.
   - Créer un vecteur `signal` contenant $100L$ points valant tous 0.
   - Définir a=[0 0 0 0 0 0] ; et b=[0 0 0 0 0 0] ;

2. Les vecteurs a et b sont les 6 premières composantes de la série de Fourier. Ajouter au script une boucle permettant de générer une série de Fourier tronquées au 6 premiers coefficients (équation 2.1).

3. Choisir les vecteurs `a` et `b` au hasard et afficher le signal en fonction de time.

4. Essayer les vecteurs $a_n = 0$ et $b_n = (-1)^{n+1}\frac{2}{\pi n}$ et afficher le signal en fonction de time. Qu'en dites vous ?

5. Ajouter maintenant 2 termes dans a et b (toujours $a_n = 0$ et $b_n = (-1)^{n+1}\frac{2}{\pi n}$) et afficher le signal en fonction de time. A-t-il changé ? Pourquoi ?

6. Réaliser une boucle pour ajouter un nombre arbitraire de coefficients dans la série et faire un test avec $a_n = 0$ et $b_n = (-1)^{n+1}\frac{2}{\pi n}$. A partir de combien de coefficients l'approximation d'un signal triangulaire vous semble bonne ?

7. Trouver la décomposition de Fourier d'une fonction *créneau* et réaliser un script afin de voir l'effet du nombre de coefficients sur la qualité de l'approximation.

Exercice 2.4 : Synthé Matlab

Dans cet exercice nous allons réaliser un synthétiseur Matlab ! Pour ce faire nous allons procéder comme à l'exercice précédent en définissant les composantes fréquentielles à ajouter dans notre signal. Puis nous utiliserons la fonction `sound` pour jouer ce magnifique morceau de musique électronique... C'est la première fois que nous allons développer un petit projet Matlab qui utilisera plusieurs fonctions, soyez attentifs à vos noms de fichier.

1. Créer une fonction qui prendre en paramètres d'entrée : T la durée du signal généré et $freq$ la fréquence de la composante. Cette fonction donnera en sortie un vecteur qui sera le signal généré.

2. La fréquence d'échantillonnage de Matlab par défaut pour le son est 8192 points par seconde. On définit donc une variable `time` telle que cette variable soit un vecteur régulièrement espacé entre 0 et T avec une longueur 8192*T. On crée aussi une variable `signal` de même taille et de valeur 0.

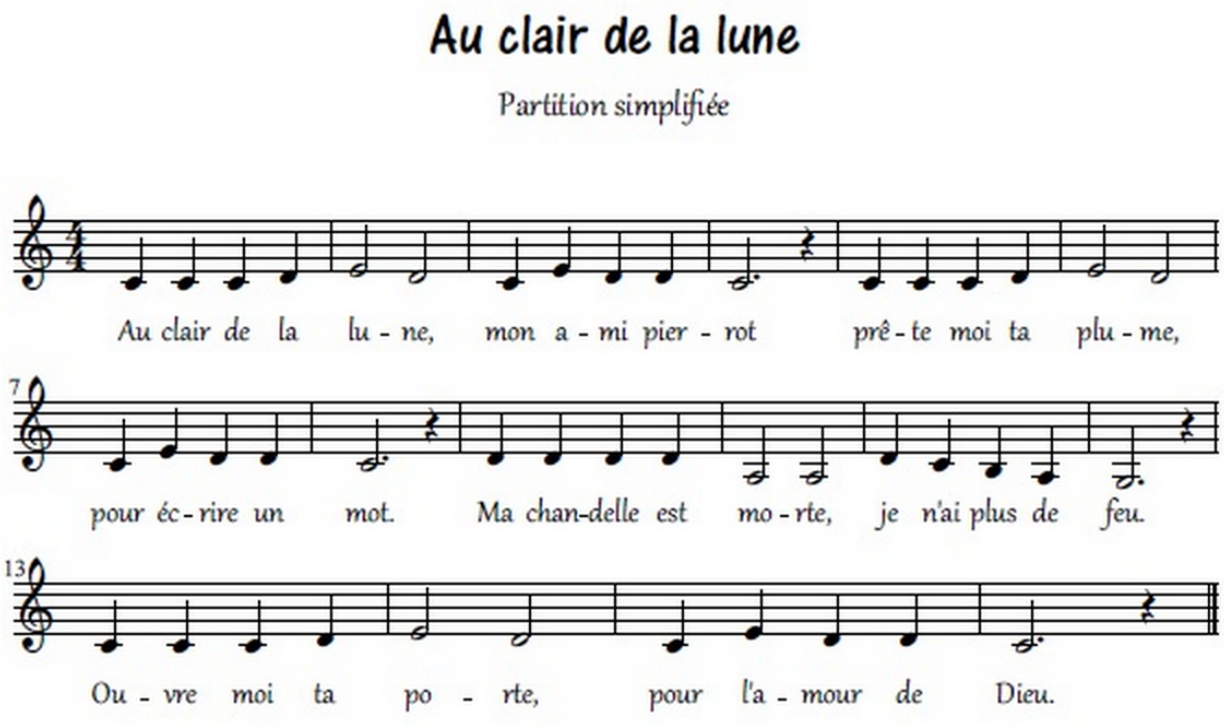
3. Créer un vecteur a de dimension égale à la fréquence maximale $freq$ et dont tous les coefficients valent 0 sauf le coefficient $freq$ qui vaut 1.

4. Réaliser une boucle qui permet de faire la synthèse du signal (equation 2.1). Attention pour avoir les fréquences en Hz et le temps en seconde, il est important d'avoir ici l'expression $\cos(n * time * 2\pi)$ dans la série de Fourier.

5. Tester le signal avec la commande `sound(...)`. PAS plus long que 2 secondes s'il vous plait !

6. Modifier le programme pour qu'il prenne en entrée non pas une valeur de fréquence mais un vecteur contenant plusieurs fréquences.

7. Faire un script qui appelle votre fonction afin de jouer "Au clair de la lune".



2.3 Transformée de Fourier

2.3.1 Définition

Nous venons de voir qu'il était possible de décomposer (ou synthétiser c'est la même chose), la plupart des signaux analytiques à l'aide des séries de Fourier. Il s'agit d'une somme des différents coefficients a_n et b_n si l'on utilise la forme trigonométrique ou c_n si l'on utilise la forme exponentielle. Lorsque l'on passe du discret (somme) au continu (intégrale), on dit que l'on réalise une transformée de Fourier. Il faut bien comprendre que l'on parle ici presque du même animal mathématiques, la principale différences étant que l'on ne réalise plus une somme des différentes composantes. Par analogie avec la relation (2.2), on écrit pour une fonction f sa décomposition de Fourier :

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k)\, \mathrm{d}k. \tag{2.4}$$

Les composantes de Fourier se retrouvent donc dans la fonction $F(k)$. Pour obtenir les coefficients de la décomposition, c'est à la fonction $F(k)$, on fait l'opération :

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x)\, \mathrm{d}x. \tag{2.5}$$

C'est cette opération qui s'appelle la transformée de Fourier. La transformée de Fourier possède de nombreuses propriétés très intéressantes, et je vous suggère de passer quelque temps avec un bon livre de Mathématiques pour les revoir...

2.3.2 Transformée de Fourier discrète

On vient de voir que pour passer de la série de Fourier à la transformée de Fourier on passait du discret au continu. Or, vous le savez maintenant, Matlab (et les ordinateurs en général) travaillent avec des données discrètes ! Il faut donc utiliser utiliser un algorithme pour faire l'opération "transformée de Fourier discrète".

La bonne nouvelle est que Cooley et Tukey dans le milieu des années 60 ont trouvé un algorithme très efficace pour faire des transformée de Fourier qui s'appelle FFT (Fast Fourier Transform). Cette algorithme repose sur une division du problème en 2 sous-problèmes, eux même divisés en 2 sous-sous-problèmes, etc. Il faut donc un nombre de point en 2^N pour que l'algorithme FFT s'applique bien, ce qui est une condition importante lorsque vous travaillerez avec la commande Matlab `fft`. Voyons maintenant comment utilise-t-on cette commande.

2.3.3 Un exemple pas à pas de FFT

Réalisons ensemble le code suivant :

```
clear all; close all; clc;

L=20;
n=128;
x2=linspace(-L/2,L/2,n+1); x=x2(1:n);

u=exp(-x.^2);
plot(x,u)

ut=fft(u)
plot(ut) % sur certaine version de Matlab il faut faire plot(real(ut)).
```

Ajouter maintenant la commande `fftshift` :

```
uts=fftshift(ut)
plot(abs(uts))
```

Il faut maintenant redéfinir correctement l'axe des fréquences :

```
k=(2*pi/L)[0:n/2-1 -n/2:-1];
```

`2pi/L` permet de normaliser au domaine L et non pas au domaine 2pi.

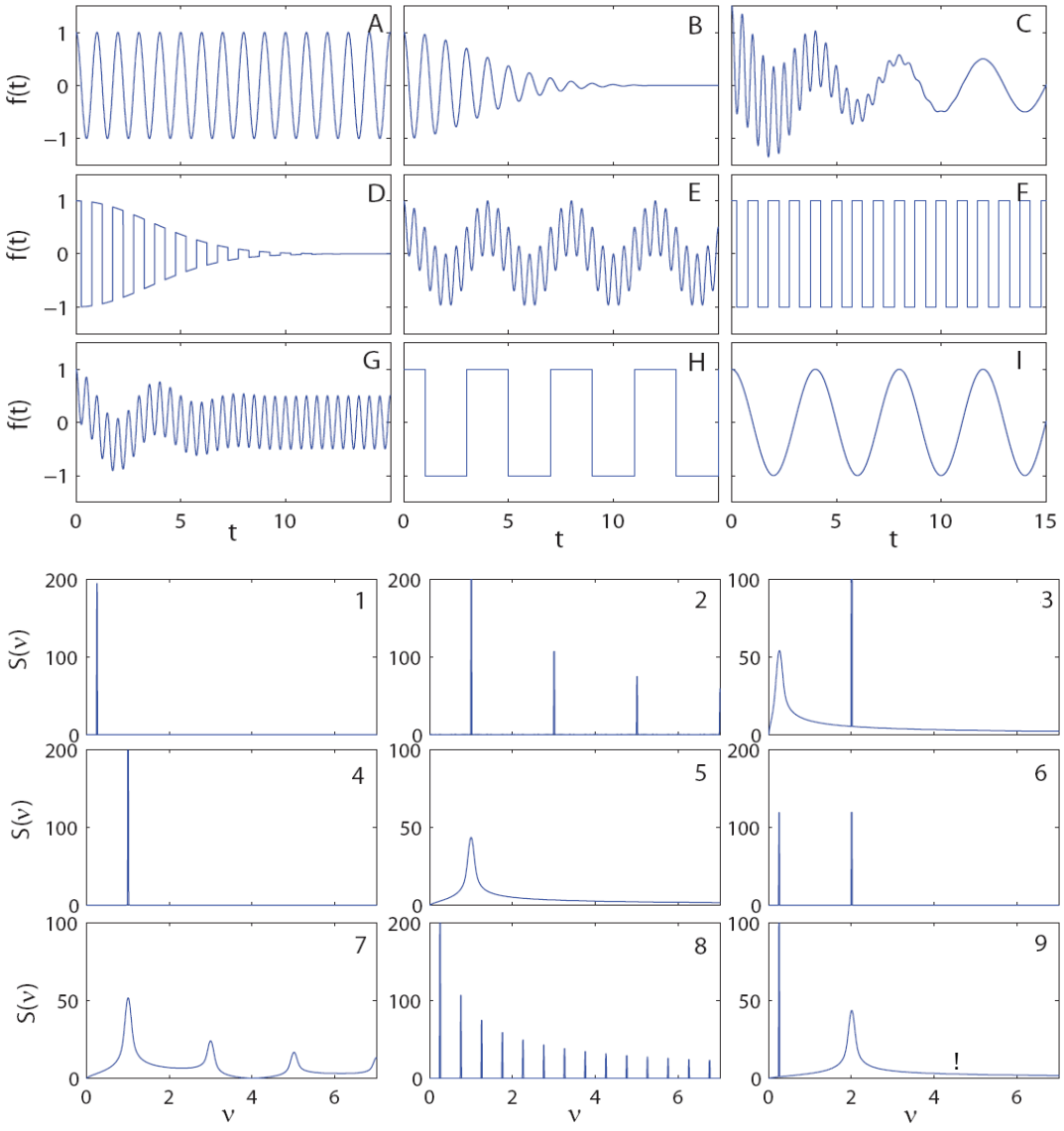
Exercice 2.6 : FFT d'un signal electrique - Rapport signal sur bruit.

On a mesuré le signal électrique $U(t)$ (tension en fonction du temps) avec un échantillonnage temporel $dt = 0.1$ ms pour des valeurs de t comprises entre 0 et 10 s. La tension $U(t)$ mesurée est donnée dans la matrice U de dimensions (100001,1). Elle contient une composante périodique et un bruit aléatoire.
Télécharger ce fichier depuis `http://quentinglorieux.fr/matlab/files/U.mat`

1.Tracer $U(t)$.
2.Tracer le spectre $S(f)$, c'est à dire le carré de la norme de la transformée de Fourier, exprimé en fonction de la fréquence f pour la fonction $U(t)$.
3. Quelle est la période de ce signal ?
4. Calculer le rapport signal sur bruit. Ce rapport est défini comme $\frac{I_1}{I_2 - I_1}$, avec I_1 l'intégrale de S pour f entre 39.9 et 40.1 Hz et I_2 l'intégrale de S pour f entre 0 et 5000 Hz.

Exercice 2.7 : Identification des transformée de Fourier

Associer chaque fonction A, B, C... à sa transformée de Fourier 1, 2, 3 ... en justifiant par les propriétés de la fonction.
Exemple : G – 3 La fonction G est une somme d'une sinusoïde de période 4 amortie sur un temps caractéristique de 5 et d'une sinusoïde de période 0,5. Le spectre 3 : un pic de fréquence 0,25 de largeur environ 0,2 et un pic étroit de fréquence 2.



Problème 2.8 : Filtrage sonore

Nous allons analyser spectralement des enregistrements sonores. On pourra créer, pour gagner du temps, la fonction TFourier.m suivante :

```
function [Nu,F] = TFourier(f,dt)
N = max(size(f));                % On détermine le nombre de points de la fonction f(t)
Nu = (0:N-1)/(N*dt);             % Rappel : Ttotal est donné par N*dt
F = abs(fft(f));                 % On prend la valeur absolue de la TF.
```

On peut ouvrir un fichier .wav dans Matlab en utilisant l'icône « ouvrir » de l'espace de travail (ou avec la commande wavplay). Ce fichier consiste en une matrice à une ligne, correspondant à un signal $S(t)$ enregistré à la fréquence d'échantillonnage `freq` c'est-à-dire que $S(t)$ est connu pour des valeurs de t séparées de $1/freq$. On peut l'écouter à la fréquence d'échantillonnage `freq` par la commande `wavplay(W,freq)`. Par défaut (si on tape `wavplay(W)`) la fréquence d'échantillonnage est fixée à la valeur standard `freq` = 11025 Hz.
Télécharger les fichiers « flute.wav » et « violon.wav » depuis `http://quentinglorieux.fr/matlab/files/`

Question 1.
Une onde sonore sinusoïdale de fréquence 784 Hz correspond à la note Sol. Ouvrir les fichiers « flute.wav » et « violon.wav », où la note Sol, jouée par un instrument, est enregistrée à la fréquence d'échantillonnage 10000 Hz.
a- Combien de temps dure chaque enregistrement ?
b- Tracer la transformée de Fourier de chaque enregistrement, en faisant attention à l'axe des abscisses. Décrire ces courbes : pic principal à 784 Hz, harmoniques, dédoublement des pics, pics parasites à basse fréquence etc.
c- Tracer la courbe de l'enregistrement de flûte entre 250 et 350 ms. Comment les différentes caractéristiques observées sur la transformée de Fourier se retrouvent-elles sur le signal ?

Conclusion : c'est notamment la présence d'harmoniques qui fait la différence, pour une même note (une même fréquence), entre deux instruments.

Question 2.
a- Par quelle commande peut-on générer artificiellement un signal sonore Wsol d'une durée de 2 secondes, à la fréquence d'échantillonnage 11025 Hz, de la note Sol ?

b- Tracer la transformée de Fourier de ce signal.

c- Si une note correspond à une fréquence donnée (par exemple, Sol : 784 Hz), la fréquence double correspond à la même note mais à l'octave suivante. Qu'obtient-on si on écoute le signal sonore Wsol à la fréquence d'échantillonnage 22050 Hz ? et à la fréquence 5512 Hz ?

Question 3.
On souhaite filtrer les hautes fréquences du signal « flute ». Pour cela, on calcule la transformée de Fourier, on supprime la partie contenant les fréquences hautes, puis on effectue la transformée de Fourier inverse pour obtenir le signal filtré :

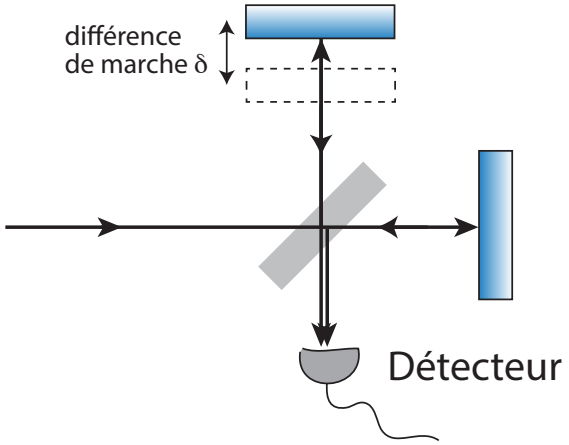
```
F = fft(Wflute); F(300:7200)=0; Wfiltre = real(ifft(F));
```

Commentez le code ci-dessus puis tracer le signal Wfiltre obtenu et sa transformée de Fourier. Montrer que l'on a réussi à supprimer la composante à 800 Hz et à faire ressortir la composante à 150 Hz.

Question 4.
On souhaite générer artificiellement la note de musique jouée par un violon. Pour cela, repérer sur le spectre du « Sol » du violon la hauteur des harmoniques 1 à 6, que l'on notera A_1 à A_6, et créer pour f = 784 Hz le signal $A_1\cos(2\pi ft) + A_2\cos(4\pi ft) + A_3\cos(6\pi ft) + A_4\cos(8\pi ft) + ...$
Faire de même pour la flûte. Ecouter ces deux signaux : a-t-on correctement reconstruit les sons des instruments ?

Problème 2.9 : Spectroscopie de Fourier (exercice d'examen 2010)

On cherche le spectre d'un signal lumineux. Ce spectre est noté $S(k)$ avec k le nombre d'onde défini comme $k = \frac{1}{\lambda}$ et exprimé en μm^{-1} .
On envoie le signal lumineux dans un interféromètre de Michelson (on rappelle son schéma ci-dessous à titre indicatif mais sa compréhension n'est pas nécessaire pour l'exercice). L'expérience consiste en la mesure de la puissance lumineuse P en sortie de l'interféromètre à l 'aide d'un détecteur (typiquement une photodiode) pour différentes valeurs de la différence de marche δ. On donne le résultat $\Delta P = P(\delta) - P(\delta = +\infty)$ de cette mesure (en unité arbitraire) dans la matrice `deltaP` en fonction de δ (en μm) dans la matrice `delta`. Les matrices sont à charger sur `http://quentinglorieux.fr/matlab/files/`.



Question 1.
Tracer $\Delta P(\delta)$ et placer les légendes.

Question 2.
La technique de spectroscopie par transformée de Fourier repose sur le fait que le spectre recherché $S(k)$ n'est autre que la norme de la transformée de Fourier de la fonction $\Delta P(\delta)$ mesurée. Tracer donc $|S(k)|$.

Question 3.
Combien cette courbe comporte-t-elle de pics ? En quelles longueurs d'onde sont-ils situés ? Lesquels vous paraissent significatifs, lesquels attribuez-vous à des artefacts ?

Question 4.
Quelle est la résolution sur k (c'est-à-dire l'échantillonnage dk entre deux valeurs de k successives) que nous pouvons atteindre dans ces conditions expérimentales ? Qu'aurait-il fallu changer dans l'expérience pour améliorer la résolution ?

Question 5.
On sait que :

$$\cos(k_1 x) + \cos(k_2 x) = 2\cos\frac{(k_1 + k_2)x}{2}\cos(\frac{(k_1 - k_2)x}{2}).$$

Quel lien peut-on faire entre la présence d'un « doublet » (deux pics très proches) dans la transformée de Fourier et la présence de « battements » (modulation des oscillations par une enveloppe sinusoïdale) de $\Delta P(\delta)$?

2.4 Corrigés

Exercice 2.3 - Correction

On crée un signal en fonction de la variable time définit à l'aide de ses composantes de Fourier. On affiche ici les 10 premières composantes.

```
L=20;
time=linspace(0,L,100*L);
signal=zeros(100*L,1)';

a=[1 1 0 0 0 0 0 0 0 0 ];
b=[0 0 0 0 0 0 0 0 0 0 ];

for n=1:length(a)
    signal=signal+a(n)*cos(n*time)+b(n)*sin(n*time);
end

plot(time,signal)
```

On affiche ensuite la série de Fourier avec $a_n = 0$ et $b_n = (-1)^{n+1}\frac{2}{\pi n}$, avec les 6 premiers coefficients.

```
L=20;
time=linspace(0,L,100*L);
signal=zeros(100*L,1)';

for n=1:6
    signal=signal+(-1)^(n+1)*2/(pi*n)*sin(n*time);
end

plot(time,signal)
```

Puis on trace la même courbes avec les 20 premiers coefficients.
Voila ce que l'on obtient :

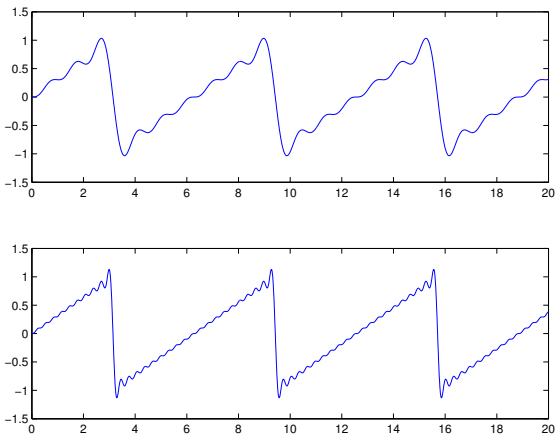


FIGURE 2.1: Séries de Fourier avec $a_n = 0$ et $b_n = (-1)^{n+1}\frac{2}{\pi n}$ pour 6 (haut) et 20 (bas) coefficients.

Enfin, voila la décomposition d'une fonction créneau :

```
for n=0:10
    signal=signal+(-1)^(n)*2/((2*n+1)*pi)*cos((2*n+1)*time);
end
```

Exercice 2.4 - Correction

Dans un premier temps, on cherche à synthétiser un son à 440 Hz. Voila le code permettant de faire cette opération sous Matlab.

```
%  SYNTHESE SONORE
clear all; clc;
L=1;
time=linspace(0,L,8192*L);
signal=time; signal=0;

a=zeros(1000,1);
b=zeros(1000,1);
a(440)=1; %choix de la fréquence ici

for n=1:length(a)
    signal=signal+a(n)*cos(n*time*2*pi)+b(n)*sin(n*time*2*pi);
end
signal=signal/max(abs(signal));
plot(time,signal)
sound(signal)
```

On peut transformer ce code relativement simplement en une fonction qui prenne en entrée les paramètres `T` et `freq` :

```
function [signal]=note(T,freq)
L=T;
time=linspace(0,L,8192*L);
signal=time; signal=0;

a=zeros(freq,1);
b=zeros(freq,1);

a(freq)=1;

for n=1:length(a)
    signal=signal+a(n)*cos(n*time*2*pi)+b(n)*sin(n*time*2*pi);
end
signal=signal/max(abs(signal));
sound(signal)
```

Et voila comment modifier le code pour en faire une fonction acceptant un vecteur pour T et pour freq. Par exemple `note([0.5 0.5 0.5],[400 800 400]);`.

```
function [signal]=note(T,freq)

for i=1:length(T)
L=T(i);
time=linspace(0,L,8192*L);
signal=time; signal=0;
a=zeros(freq(i),1);
b=zeros(freq(i),1);
a(freq(i))=1;
for n=1:length(a)
    signal=signal+a(n)*cos(n*time*2*pi)+b(n)*sin(n*time*2*pi);
end
signal=signal/max(abs(signal));
sound(signal)
pause(T(i))
end
```

Exercice 2.6 - Correction

Après avoir charger le fichier on affiche la fonction U, puis la norme carrée de sa transformée de Fourier (son spectre).

```
plot (U)
Ut=abs(fft(U)).^2;
k=(1/10)*[0:100000];
plot(k,Ut)
axis([1 100 0 5e7])
```

On observe alors un pic autour de 40 Hz, c'est à dire 25 ms de période.
On calcule ensuite le rapport signal sur bruit (RSB) :

```
signal = Ut(401);
bruit = sum(Ut([1:50001])) - signal;
RSB=signal/bruit
```

On applique ensuite un filtre gaussien pour éliminer le bruit :

```
center=40;
sigma=5;
gauss=exp(-(k-center).^2/sigma^2);
subplot (3,1,2); plot(k,gauss)
axis([1 100 0 1])

subplot (3,1,3); plot(k,gauss.*Ut)
axis([1 100 0 1e7])
```

Voila ce que l'on obtient à l'aide du filtre gaussien :

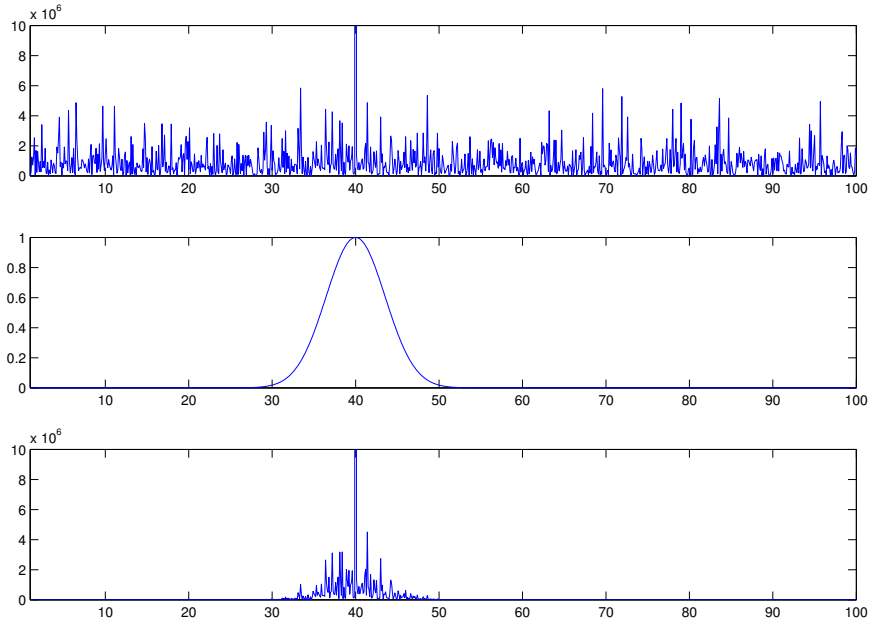


FIGURE 2.2: De haut en bas, on obtient le spectre brut, le filtre gaussien, et le spectre filtré.

On peut alors effectuer la TF inverse.

```
signalf=ifft(gauss.*Ut);
figure(2)
subplot (2,1,2);plot (real(signalf))
axis([1 1000 -1e5 1e5])
subplot (2,1,1);plot (U)
axis([1 1000 0 15])
```

Voila le signal filtré :

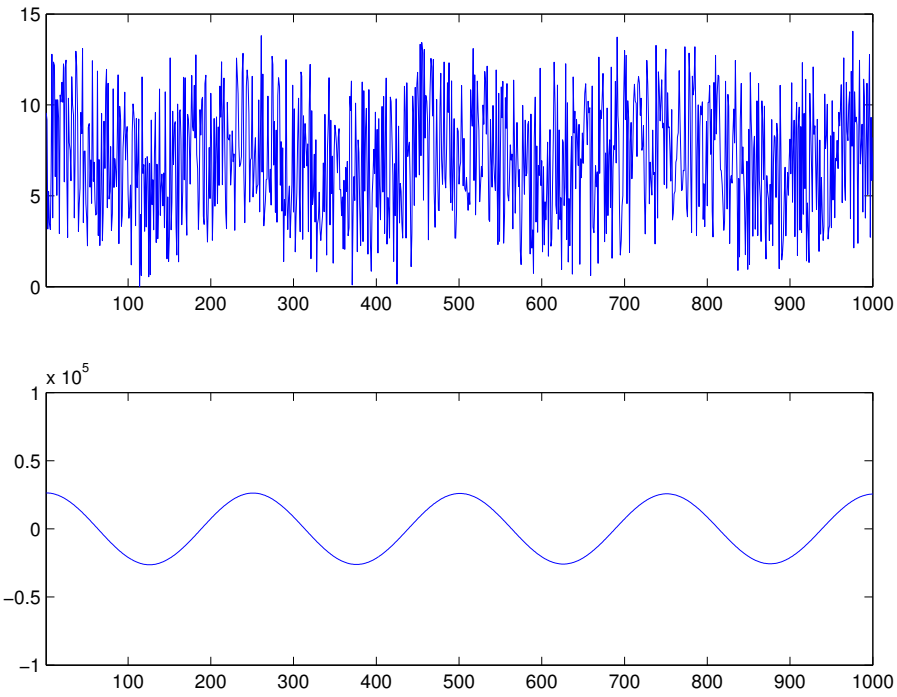


FIGURE 2.3: De haut en bas, on obtient le signal brut, et le signal filtré.

Exercice 2.7 - Correction

Résumons les propriétés des différentes fonctions dans un tableau (on distingue une composante périodique 1 et une éventuelle composante périodique 2) :

Fonction	A	B	C	D	E	F	G	H	I
période 1	1	1	0,5	1	0,5	1	0,5	4	4
amortissement 1	infini	5	5	5	infini	infini	infini	infini	infini
1 sinusoïdale ?	Non	Non	Non	Oui	Non	Oui	Non	Oui	Non
période 2	-	-	4	-	4	-	4	-	-
amortissement 2	-	-	infini	-	infini	-	5	-	-
2 sinusoïdale ?	-	-	Non	-	Non	-	Non	-	-

Par ailleurs on a les transformées de Fourier suivantes (on distingue les pics 1 et 2) :

T. Fourier	1	2	3	4	5	6	7	8	9
fréquence 1	0,25	1	2	1	1	2	1	0,25	2
largeur 1	0	0	0	0	0,2	0	0,2	0	0,2
Harmoniques de 1 ?	Non	Oui	Non	Non	Non	Non	Oui	Oui	Non
fréquence 2	-	-	0,25	-	-	0,25	-	-	0,25
largeur 2	-	-	0,2	-	-	0	-	-	0
Harmoniques de 2 ?	-	-	Non	-	-	Non	-	-	Non

Pour les temps d'amortissement et les largeurs des pics, on sait que l'un est de l'ordre de l'autre, mais on ne donne ici que des ordres de grandeur ; pour donner des valeurs plus précises, il faudrait avoir défini précisément la notion de largeur : par exemple « largeur à mi-hauteur » etc

Sachant qu'une composante de période T, amortie sur un temps T_2, correspond à un pic à la fréquence 1/T de largeur $1/T_2$ on conclut :

Fonction	A	B	C	D	E	F	G	H	I
T. Fourier	4	5	9	7	6	2	3	8	1