Signaux et systèmes 1

Travaux de laboratoire

8 octobre 2008

.

Table des matières

Liste des exemples

Liste des figures

Introduction

Les études de laboratoire sont l'occasion de développer le goût du concret, le sens de l'efficacité, l'esprit d'initiative et la confiance en un savoir faire. Ces études donnent lieu à un compte-rendu synthétique, correctement rédigé, dans lequel figurent les éléments significatifs et les commentaires nécessaires montrant une bonne maîtrise du sujet traité. Les élèves sont par ailleurs invités à participer à l'orientation de leur étude, cette participation étant croissante avec le nombre de séances dévolues à un même sujet. Un compte-rendu sera demandé à l'issue de l'ensemble des séances de travail.

Rédaction du compte-rendu

Vous trouverez aussi quelques conseils à ce sujet sur `http://moodle.supelec.fr`, dans le Guide de Rédaction du Document Ecrit, dans la rubrique Travaux de Laboratoire du cours « Signaux et Systèmes 1 ».

- Il n'est pas obligatoire de rédiger le compte-rendu à l'aide d'un outil de traitement de texte, même si la forme informatique est conseillée. Dans ce cas, une version papier sera remise aux correcteurs.
- Chaque étape de l'étude sera précédée d'une introduction (qu'est-ce que l'on cherche ?), illustrée par des figures judicieusement choisies **insérées au fil du texte (les feuilles volantes jointes ne seront pas prises en compte)** et des commentaires pertinents (qu'est-ce qu'on obtient ?), et suivie d'une conclusion également pertinente (pourquoi ?)
- Tous les résultats présentés dans le compte-rendu, en particulier les figures pourront être générées à la demande à partir d'un fichier de commandes autosuffisant.

Développement Matlab

- Toute fonction `ma_fonction.m` développée devra être commentée comme les fonctions internes du logiciel, en particulier `help ma_fonction` affichera les informations suivantes :
  - Format de la ligne de commande
  - Entrée(s) : nature et valeurs
  - Sortie(s) : nature et valeurs
- Des exemples de fichiers de commandes et de fonctions seront fournis dans le compte-rendu.

– Les fichiers Matlab développés seront regroupés dans un dossier sur votre compte et devront rester à la disposition des correcteurs jusqu'à la remise des notes.

Notation de l'étude

La notation de l'étude tiendra compte des critères suivants :

– **compte-rendu**
  • clarté et pertinence des explications ;
  • analyse et interprétation des résultats ;
  • choix et présentation judicieux des informations sur les graphiques (en particulier, axes correctement gradués, échelles, superposition. . .) ;
  • qualité des logiciels développés.
– **travail fourni**
  • conformité aux objectifs précisés pour chaque thème dans le texte de l'étude ;
  • quantité et qualité des résultats.
– **comportement en séance**
  • initiative et créativité de chacun ;
  • qualité du travail en binôme ;
  • gestion du temps pour atteindre les objectifs.

Partie 1

Quelques opérations de base sur les signaux

De nombreux exemples de code vous sont fournis en annexe de ce document, ils sont volontairement peu ou pas commentés pour vous apprendre à utiliser l'aide de Matlab. Vous pouvez :

- soit taper `help instruction` depuis la fenêtre de commande si vous connaissez le nom de l'instruction à utiliser
- soit ouvrir la fenêtre d'aide pour une recherche plus approfondie (mais parfois plus compliquée !)

Essayez de bien comprendre ces exemples et pensez à utiliser par la suite ce que vous avez appris.

Commencez par recopier les fichiers du `.zip` se trouvant sur `http://moodle.supelec.fr`, dans la rubrique « Travaux de Laboratoire »de Signaux et Systèmes 1 (cours d'Olivier Pietquin) dans votre répertoire de travail sur votre compte. Il s'agit des fichiers d'exemples au format `.m` et de signaux sonores au format `.wav`. Sous `MATLAB`, réglez alors le chemin d'accès vers ce répertoire (`current directory` dans la barre d'outils).

1.1 Représentation, acquisition et restitution de signaux

Vous disposez des fichiers `sinus1.wav` et `sinus2.wav`. Pour récupérer à la fois le signal enregistré dans le fichier `sinus1.wav` et sa fréquence d'échantillonnage, que vous appellerez respectivement `sig1` et `fe`, utilisez la fonction `wavread` en choisissant la syntaxe correcte. Tracez ce signal en fonction du temps en graduant l'axe des abscisses en secondes, en vous aidant de l'exemple B.1 page 19, pour lequel vous expliquerez l'effet de l'instruction `plot`.

Représentez les signaux des fichiers `sinus1.wav` et `sinus2.wav` sur une même figure en respectant l'échelle des temps, en vous inspirant des trois exemples B.2 page 20.

Lancez le logiciel `Audacity` (raccourci sur `Ntelev` dans le dossier `Winapp`), enregistrez en mono le signal de sortie du microphone lorsque vous prononcez deux mots distincts, en utilisant une fréquence d'échantillonnage f_e de 11 kHz environ (réglage dans le coin inférieur gauche de la

fenêtre, valeur par défaut : 44 100 Hz) et sauvez-le au format `.wav`. Récupérez ce signal sous Matlab, tracez-le en fonction du temps en graduant l'axe des abscisses en secondes. Ecoutez-le en utilisant la fonction `sound` à différentes fréquences de restitution : que peut-on dire sur la durée et le contenu spectral des enregistrements reproduits selon la valeur de cette fréquence ?

1.2 Manipulation de signaux numériques

Faites une extraction de chaque mot du fichier précédent (voir l'exemple B.3 page 21 et créez deux nouveaux fichiers wav contenant chacun un de ces mots (fonction `wavwrite`). Vérifiez que vous avez correctement réalisé l'opération en écoutant ces signaux sous le Media Player de Windows par exemple.

Calculez le plus simplement possible l'énergie de chacun de ces trois signaux, sachant que `x'` est le vecteur `x` transposé. Il est ici strictement interdit d'utiliser une boucle `for` !

De l'utilité des scripts Mauvaise nouvelle pour vous : si votre PC plante maintenant, vous aurez perdu tout le travail que vous avez effectué jusqu'à présent. Aussi, pour éviter ce genre de désagréments, créez un `M_File`, fichier qui permet de sauvegarder une suite d'instructions sous `MATLAB` (menu `File New M-File`), recopiez-y toutes les commandes correctes utilisées jusqu'à présent et enregistrez-le. Vous pouvez exécuter ce fichier en tapant son nom dans la fenêtre de commande. **Vous sauverez au fur et à mesure toutes vos instructions utiles sous cette forme dans la suite de l'étude.**

Partie 2

Classification des signaux

La fonction d'autocorrélation d'un signal peut servir à déterminer sa classe. Vous commencerez par programmer cette fonction avant de passer à son utilisation en classification.

2.1 Exemple de calcul théorique

Quelle est la fonction d'autocorrélation d'un signal sinusoïdal pur $y(t) = A\sin(2\pi ft)$?

2.2 Programmation

En vous inspirant de l'exemple B.4 page 22, créez une fonction renvoyant l'autocorrélation d'un signal, sans utiliser l'instruction `xcorr`. Votre fichier devra être correctement commenté, comme dans l'exemple fourni. Vous écrirez un script permettant de tester votre fonction sur l'un des sinus utilisés précédemment et de comparer votre résultat à celui de la fonction `xcorr` grâce à un tracé sur le même graphique. Obtenez-vous ce qui est attendu et si non, pourquoi ? Vous modifierez ensuite votre fonction d'autocorrélation afin qu'elle renvoie directement les abscisses correctes pour le tracé de la courbe.

2.3 Application à la classification de quelques signaux simples

A l'aide du logiciel `OlCom` et du générateur de signaux (cf. exemple B.5 page 23), faites l'acquisition des signaux suivants :

- un signal sinusoïdal de fréquence 200 Hz (sur quelques périodes)
- un signal triangulaire (idem)
- un signal de bruit (issu du boîtier nommé générateur de bruit)

Enregistrez également sous `Audacity` la syllabe « aa » sur une durée d'environ 1 seconde.

Tracez l'autocorrélation de plusieurs tranches du « aa » durant quelques dizaines de millisecondes, du signal de bruit et de quelques périodes des signaux sinusoïdal et triangulaire. Ces signaux sont-ils stationnaires ou non ?

2.4 Classification de signaux de parole voisés ou non voisés

Vous trouverez dans l'annexe A quelques généralités sur le signal de parole. Enregistrez le mot « chat » en le prononçant lentement : « chhhhhhhh-aaaaaaaaaa », en choisissant une fréquence d'échantillonnage de 8 ou 16 kHz. Calculez l'autocorrélation de différentes tranches de signal de durées variables et utilisez ces résultats pour découper le signal en parties voisées et non voisées. Evaluez la fréquence fondamentale sur les parties voisées. Quelques courbes judicieusement choisies illustreront vos commentaires.

Partie 3

Aspects fréquentiels

3.1 La Transformée de Fourier discrète (TFD)

La transformée de Fourier discrète est calculée sous Matlab par la fonction `fft` (pour Fast Fourier Transform en anglais), utilisant si possible un algorithme rapide sur 2^n points (TFR : Transformée de Fourier Rapide en français).

Créez une fonction Matlab permettant d'effectuer le tracé de la densité spectrale de puissance d'une tranche de signal issue d'un fichier `.wav`, avec comme variables d'entrée :

- le nom du fichier `.wav`
- la taille de la tranche
- l'indice de début de tranche

L'axe des abscisses sera gradué en fréquences (Hz). L'échelle des ordonnées sera en dB. Vous pourrez vous inspirer de l'exemple présenté dans l'aide de Matlab pour la fonction `fft`. Appliquez votre fonction à des tranches des signaux classifiés précédemment et vérifiez que les résultats sont cohérents avec ceux obtenus par la méthode temporelle. On rappelle qu'un signal périodique peut être décomposé en série de Fourier selon ses harmoniques, par exemple, pour la note de musique la_3 de hauteur 440 Hz, les harmoniques sont 880 Hz, 1320 Hz, 1760 Hz . . .L'harmonique de rang 1 (440 Hz) est appelée fondamentale : elle correspond au pitch de la note (mélodie).

Ecrivez une fonction effectuant du zero-padding sur un signal. Les entrées seront :

- le signal d'entrée
- le nombre de zéros ajoutés

et la sortie sera le signal après zero-padding. Vous pourrez avoir besoin des fonctions de concatenation de matrices comme `horzcat`. Comparez les DSP d'un signal avant et après zero-padding. Comment faire du zero-padding directement avec la fonction `fft` ? Modifiez en conséquence la fonction du tracé de la DSP en ajoutant comme entrée la taille de la transformée de Fourier.

3.2 Changement de fréquence d'échantillonnage

En vous inspirant de l'exemple B.6 page 23, écrivez deux fonctions réalisant respectivement une augmentation et une réduction de cadence sur un signal `.wav` et créant le `.wav` correspondant. Les entrées seront :

- le nom du fichier `.wav`
- la taille de la tranche
- l'indice de début de tranche
- le facteur de changement de cadence
- le nom du fichier `.wav` après changement de cadence

Les sorties seront :

- le signal après changement de cadence
- la nouvelle fréquence d'échantillonnage

Ecoutez le résultat pour une large plage de facteurs de changement de cadence et commentez les résultats. Comparez sur un même graphique les transformées de Fourier du signal d'origine et du signal après changement de cadence.

Effectuez les mêmes opérations en utilisant la fonction `resample` et expliquez les différences à l'écoute.

3.3 Analyse spectrale

3.3.1 Analyse d'une tranche de signal par TFD

Créez une fonction générant le signal `x` résultant de l'échantillonnage à la fréquence f_e du signal analogique $x(t) = A_0 \cos(2\pi f_0 t) + A_1 \cos(2\pi f_1 t)$. Les entrées seront la taille du signal de sortie, f_e, f_0, f_1, A_0 et A_1 (revoir l'exemple B.1 page 19).

Vérifiez en le traçant que le module de la transformée de Fourier d'une fenêtre de Hanning de longueur N peut être considéré comme un triangle de surface unité et de largeur $4/N$ (cf. exemple B.7 page 23).

Créez une fonction analysant une tranche de signal en calculant sa TFD, la tranche étant au préalable pondérée par une fenêtre de Hanning.

Testez vos fonctions en effectuant l'analyse d'une tranche de 256 points du signal `x` avec $A_0 = A_1 = 1$, $f_0 = 40\,\text{kHz}$, $f_1 = 61\,\text{kHz}$ et $f_e = 512\,\text{kHz}$ et retrouvez les résultats de l'exercice 2 du TD 4, dont l'énoncé est rappelé dans l'exemple B.8 page 24.

3.3.2 Effets de quelques fenêtres

Modifiez alors votre fonction pour laisser à l'utilisateur le choix de la fenêtre. Faites l'analyse d'une tranche de 256 points du signal `x` avec $A_0 = 1$, $A_1 = 10^{-2}$, $f_0 = 995\,\text{Hz}$, $f_1 = 1200\,\text{Hz}$ et $f_e = 8\,\text{kHz}$ en utilisant différentes fenêtres, par exemple celles de Hanning, de Hamming et de Blackman. Quel est l'effet du « zero-padding » ?

Partie 4

Application : détection de pitch

4.1 La prosodie : le sens d'une phrase

Prononcez cette phrase :

Toto dit : « le maître est un âne ».

Puis cette autre :

Toto, dit le maître, est un âne.

La prosodie contient les notions de variations de rythme, de durée, d'amplitude et de hauteur de la parole. On constate sur l'exemple précédent l'importance de la ponctuation dans le sens (la sémantique) de la phrase, même si les mots (la syntaxe) sont identiques. Ainsi une phrase interrogative se traduit par une hausse du ton sur le mot sur lequel se pose la question :

« Tu dé$^{\textbf{jeunes}}$ à midi ? » (déjeuner ou ne pas déjeuner, là est la question)
« Tu déjeunes à mi$^{\textbf{di}}$? » (ou à midi trente)

Il est donc nécessaire, si l'on désire saisir le sens d'une phrase et donc l'intention du locuteur grâce à un système automatique, d'identifier la prosodie en plus des phonèmes prononcés. Ceci est effectué par la détection du pitch.

4.2 La mélodie : la ligne mélodique

Lorsque un locuteur chante, il génère des notes de musique. Dans la musique occidentale, ces dernières sont définies par leur hauteur (fréquence fondamentale) et réparties selon une échelle logarithmique : le codage MIDI (Musical Instrument Digital Interface) définit la fréquence en fonction du code de la note comme suit :

$\text{La}_3 = 440\,\text{Hz}$ Code= 69

Il vient donc, puisque passer d'une note à son octave (12 intervalles) revient à doubler sa fréquence :

$$f = 440 \times 2^{\frac{n-69}{12}}$$

La note de code 0 (a donc comme fondamentale $8,18\,\text{Hz}$, elle est inaudible. La note Do_0 (24) a pour fondamentale $32,7\,\text{Hz}$, la note de code 127 (Sol_8) $12544\,\text{Hz}$, autant dire qu'aucun instrument de musique (a fortiori aucun humain) n'est capable de la générer.

Les applications de la reconnaissance de la mélodie sont nombreuses :
- Recherche automatique de morceaux dans une base de données
- Création automatique de partitions musicales
- Contrôle de la justesse d'un chant ...

4.3 Réalisation

Concevez l'algorithme du programme de traitement.

Créez des fonctions spécifiques pour les différentes étapes, détection voisé ou non voisé, recherche du pitch, détection de notes.

Créez la fonction principale, qui aura pour paramètres d'entrée :
- Le nom du fichier .wav original
- La durée de la tranche temporelle de base
- Le domaine de recherche du pitch
- La durée minimale d'une note

et comme variables de sortie :
- Une matrice à 3 colonnes : temps, fréquence, note MIDI

Essayez votre fonction sur les fichiers de parole et de chant enregistrés et les fichiers tests parole et musique fournis. Conclure.

Annexe A

Généralités sur le signal de parole

Le signal de parole est généré par l'envoi d'un flux d'air $p(t)$ en provenance des poumons dans le conduit vocal comme le montre la figure A.1. Les cordes vocales peuvent vibrer de façon périodique et donc « périodiser » le flux d'air continu à l'origine : le signal présent à cette étape est appelé l'« onde glottique ». Cette onde traverse ensuite les diverses cavités qui vont en modifier les caractéristiques spectrales. La figure A.2 présente la formation d'un signal voisé.

La fréquence fondamentale F correspondant à la période de vibration des cordes vocales est appelée « pitch » dont l'évolution au cours du temps est la « prosodie » (voix parlée) ou « mélodie » (voix chantée). Les zones d'énergie correspondant aux résonances du conduit vocal sont appelées « formants » : ils sont caractéristiques du phonème prononcé.

Les caractéristiques générales de ce signal sont les suivantes :
- Bande passante : 15 Hz à 20 kHz
- Niveau : de 20 dB (voix très basse) à 50 dB environ (47 dB : conversation à 1 m)

Les sons de parole se décomposent de la façon suivante :
- Sons voisés : voyelles /a/, /e/, /i/ ... consonnes /v/, /z/, /j/ (voisement+bruit), et /m/, /n/ (couplage nasal)
- Sons non voisés : sifflements, chuintements ... /s/, /ch/, /f/, plosives /p/, /t/, /k/, et occlusives /b/, /d/, /g/

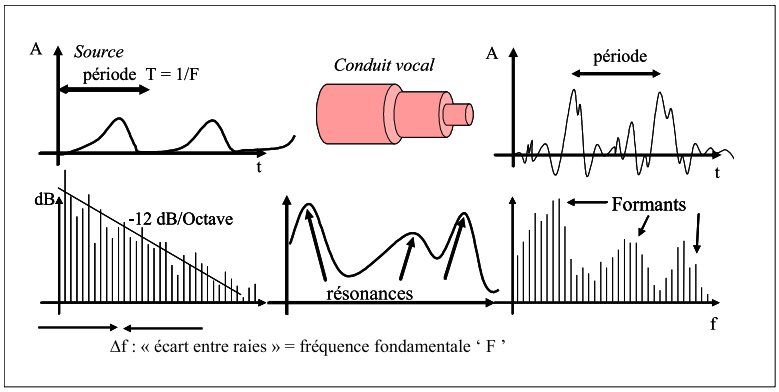

FIG. A.1 – Le conduit vocal

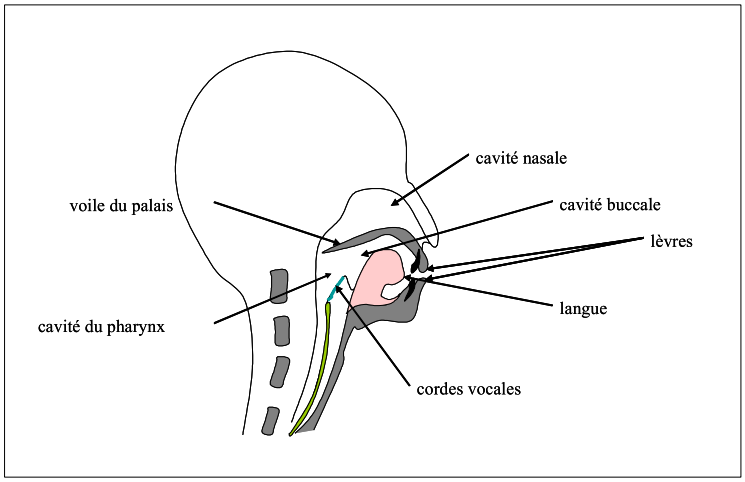

FIG. A.2 – Modèle de phonation

Annexe B

Exemples

Exemple B.1 – Tracer une courbe sous Matlab

```
% trace1.m

Te = 0.1;
N = 200;
x = 0:Te:(N-1)*Te;  % création d'un vecteur x = [0 Te 2*Te ... (N-1)*Te]
y = sin(x);

figure
plot(x,y,'+')
grid on, zoom on
xlabel('t(s)');
ylabel('valeur');
title('Signal y = sin(t)');
```

Exemple B.2 – Tracer plusieurs courbes sur une même figure

```
% trace2.m

clear all, close all
Te1 = 0.02;N1 = 100;
x1 = 0:Te1:(N1-1)*Te1;
y1 = sin(2*pi*x1);
Te2 = 0.04;N2 = 50;
x2 = 0:Te2:(N2-1)*Te2;
y2 = sin(2*pi*x2);

figure
plot(x1,y1,'.',x2,y2,'+')
grid on, zoom on
xlabel('t(s)');
ylabel('valeur');
legend('y1','y2')
title('Signaux y1 et y2');

figure
plot(x1,y1,'.-')
title('Signal y1');
hold on
plot(x2,y2,'or')
xlim([0.45 1])
hold off
grid on, zoom on
xlabel('t(s)');
ylabel('valeur');
legend('y1','y2')
title('Signaux y1 et y2');

figure
subplot(2,1,1)
plot(x1,y1,'.')
ylabel('y1');
title('Signaux y1 et y2');
grid on
subplot(2,1,2)
plot(x2,y2,'.')
grid on, zoom on
xlabel('t(s)');
ylabel('y2');
```

Exemple B.3 – Extraire des parties de signal

```
% extraire.m

clear all, close all
[x,fe] = wavread('PAROLE.wav',[121800 147500]);
figure
plot(x) % c'et une des rares utilisations de plot sans préciser d'abscisses
grid on, zoom on
xlabel('Numéro de l''échantillon')
ylabel('Valeur')
title('Extrait du signal de parole')
n1 = 13800;
n2 = 20000;
mot1 = x(1:n1);
mot2 = x(n1+1:n2);
mot3 = x(n2+1:length(x));
% Vous pouvez écouter le signal initial puis les signaux scindés
% en utilisant la fonction sound
```

Exemple B.4 – Création et utilisation de fonctions

```
% utiliser_extraction.m

clear all, close all

[x,fe] = wavread('PAROLE.wav',[121800 147500]);
indice = 20000;
[sig1,sig2] = extraction(x,indice);
```

Le script `utiliser_extraction` fait appel à la fonction `extraction`. Les lignes précédées du symbole % sont considérées comme des commentaires ; en particulier, celles qui constituent l'entête du fichier apparaissent lorsque vous utilisez l'instruction `help`. On y précise le rôle de la fonction, ainsi que ses entrées et sorties.

```
% function [res1,res2] = extraction(sig,coupure)
%
% Scinde le signal sig en deux signaux res1 et res2 à l'indice coupure
% Entrées:
%   sig: signal d'origine
%   coupure: indice de coupure
% Sorties:
%   res1 = sig(1:coupure)
%   res2 = sig(coupure+1:end)

function [res1,res2] = extraction(sig,coupure)

res1 = sig(1:coupure);
res2 = sig(coupure+1:end);
```

Exemple B.5 – Acquisition d'un signal sous OLCOM ; récupération de ce signal sous Matlab

---

- Affichez sur l'oscilloscope du logiciel OlCom (raccourci dans `Démarrer Programmes Data Translation DT300Series`) la sortie analogique $S0$ du générateur de signaux et faites les réglages pour obtenir un signal sinusoïdal pur de fréquence 200 Hz d'amplitude environ 1V vous choisirez le mode de commande **externe**). Faites une acquisition de l'ordre de quelques périodes, exportez le résultat sous le nom `sinus_200` puis enregistrez-le dans votre répertoire de travail.
- Sous `MATLAB`, tapez alors dans cette nouvelle fenêtre `sinus_200` pour créer la variable `mesure`, tableau contenant les données acquises sous forme de colonnes (ses dimensions apparaissent dans le `Workspace`).
- La première colonne de `mesure` contient les instants d'échantillonnage, les autres colonnes les valeurs correspondantes des signaux acquis.

Exemple B.6 – Changement de cadence

```
% change_freq.m

clear all, close all
[x,fe] = wavread('PAROLE.wav',[121800 147500]);
n = 2;
x1 = x(1:n:end); % x1 = [x(1) x(n+1) x(2*n+1)...]
x2 = zeros(n*length(x),1);
x2(1:n:end) = x;  % x2 = [x(1) 0 x(2) 0 x(3) 0...]
```

Exemple B.7 – Création d'une fenêtre pour l'analyse spectrale

```
% function fenetre = analyse_spec(nom_fenetre,longueur)
% Exemples d'utilisation:
% fenetre = analyse_spec(@hann,512);
% fenetre = analyse_spec(@rectwin,1024);

function fenetre = analyse_spec(nom_fenetre,longueur)

fenetre = window(nom_fenetre,longueur);
```

Exemple B.8 – Exercice 2 du TD 4

Soit le signal $x(t) = A\cos(2\pi f_0 t) + A\cos(2\pi f_1 t)$ avec $f_0 = 40\,\text{kHz}$ et $f_1 = 61\,\text{kHz}$. Le signal est échantillonné à la fréquence $f_e = 512\,\text{kHz}$. Une tranche de 256 points est ensuite analysée par un processeur calculant la TFD du signal. La tranche est au préalable pondérée par une fenêtre de Hanning.

1. Dessiner la représentation spectrale $X_{256}^{Hanning}(\nu)$ de la tranche de 256 points ainsi pondérée (on approchera la fenêtre de Hanning, dans le domaine spectral, par un triangle de surface unité et de largeur $4/N$)
2. En déduire l'estimation du module de la TFD.
