Bureaux d'étude

Traitement numérique du signal

Filière ASI, année 2019-2020

Objectifs

Ce BE a un double objectif :

1. **illustrer les concepts et les outils exposés dans le cours de traitement numérique du signal**, et plus particulièrement :
   — l'acquisition et la restitution de signaux numériques (BE1)
   — la transformée de Fourier discrète (BE2)
   — le filtrage numérique (BE3)
2. **vous amener à être capable de traiter et analyser par vous-même un signal expérimental** (BE4)

Remarques générales

Ces BE s'articulent autour de 4 sujets complémentaires :
— Acquisition et restitution de signaux (1 séance environ)
— Transformée de Fourier discrète (2 séances environ)
— Synthèse de filtres numériques (2 séances environ)
— Analyse de données expérimentales (2 séances environ)

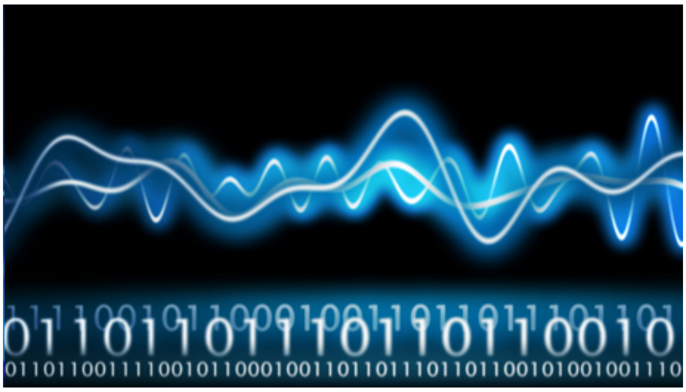
Les parties apparaissant **en bleu** peuvent être considérées comme optionnelles, et ne vous seront donc pas indispensables pour réussir votre évaluation. Toutefois, les points qui y sont abordés vous permettront de parfaire votre formation en TNS.

Enfin, vous allez travailler avec le logiciel de programmation scientifique Matlab.

Bon travail à tous !

Acquisition et restitution de signaux

environ 1 séance



L'objectif de cette première séance est de se (re)familiariser avec Matlab et de découvrir quelques notions très basiques du traitement numérique des signaux.

Matlab signifie initialement « Matrix Laboratory » et l'ensemble des fonctions qui y sont implantées en langage C utilisent des structures de vecteurs, de matrices ou de tenseurs. Les calculs sont donc d'autant plus rapides que ces structures sont utilisées dans vos codes. Ainsi, la plupart du temps, on aura intérêt à éviter l'usage de boucles (`for ... end`) au bénéfice d'opérations vectorielles ou matricielles.

1 Simulation numérique d'un signal continu

On se propose de simuler un signal continu à l'aide d'une série numérique. Pour cela on considère un signal qui serait très fortement suréchantillonné par rapport à son contenu fréquentiel.

- Créer un signal contenant 3 sinusoïdes d'amplitudes respectives 1, 2 et 3 Volts, de fréquence 300, 500 et 400 Hz, de phases initiales aléatoires, et échantillonné à $f_a = 10^5$Hz.
  (fonctions Matlab utiles : `sin, rand`)
- Visualiser les 1500 premiers points de ce signal « continu » en faisant apparaître les grandeurs physiques sur chaque axe.
  (`figure, plot, xlabel, ylabel, title, hold`)

2 Acquisition de signaux

L'acquisition d'un signal continu se réalise en trois étapes successives qui seront étudiées dans cette section :

1. échantillonnage du signal continu
2. blocage du signal échantillonné
3. quantification du signal bloqué

2.1 Échantillonnage

On souhaite tout d'abord échantillonner le signal continu précédent à la fréquence $f_e = 5$ kHz.

— Représenter sur un même graphique le signal continu et les échantillons correspondants, pour la même fenêtre d'observation que précédemment.
  (`stem`)

2.2 Blocage

Lors du blocage d'ordre zéro, le signal échantillonné est maintenu entre 2 échantillons successifs à une valeur constante égale à celle du dernier échantillon.

— Créer la fonction `Hold_B0` suivante dans un fichier dont le nom sera obligatoirement `Hold_B0.m`.

```
function SampleHold = Hold_B0(SigEch, r)
[nrows, ncols] = size(SigEch);
if nrows > ncols,
    SigEch = SigEch';
end
SampleHold = ones(r, 1)*SigEch;
SampleHold = reshape(SampleHold, 1, r*length(SigEch));
```

  Cette fonction permet simplement de répéter chaque échantillon de son signal d'entrée r fois.

— Employer cette fonction sur le signal précédemment échantillonné pour simuler le fonctionnement d'un bloqueur d'ordre zéro et obtenir son signal de sortie, puis représenter les signaux original, échantillonné et bloqué sur une même figure. Interprétez les résultats.

2.3 Quantification

Pour étudier les conséquences de la quantification, on introduit un pas de quantification de valeur q Volts. De plus, le quantifieur considéré est supposé posséder une dynamique infinie, ce qui permet de négliger les effets de saturation. Pour réaliser la quantification d'un signal s par un pas de quantification q, on pourra employer le code suivant : `sq = q*round(s/q);`.

— Le quantifieur ne fournissant à sa sortie qu'une seule valeur numérique par période d'échantillonnage, on l'appliquera ici directement au signal échantillonné. Représenter sur un même graphique les échantillons du signal avant et après quantification pour $q = 0.5$, ainsi que les échantillons de l'erreur de quantification obtenue.

— Évaluer pour $q = 2, 1, \frac{1}{2}, \ldots, \frac{1}{2^{14}}$ la variance des erreurs de quantification obtenues.
  (`var`).

— Comparer l'évolution de la variance d'erreur de quantification en fonction de q avec la valeur théorique vue en cours. On aura intérêt à utiliser un diagramme logarithmique pour mettre en évidence la loi de puissance de cette relation.
  (`loglog`)

3 Restitution de signaux

Le but est ici d'étudier l'opération de « restitution analogique », c'est à dire l'opération permettant de générer l'équivalent continu d'un signal numérique donné. Cette opération se réalise le plus souvent en deux étapes :

1. application d'un bloqueur d'ordre zéro sur le signal numérique à restituer
2. filtrage passe-bas du signal bloqué pour atténuer les effets de « marche d'escalier »

— Repartir de la version échantillonnée et quantifiée avec $q = 0.5$ du signal original. Appliquer à ce signal numérique le bloqueur d'ordre zéro employé précédemment.

— Synthétiser un filtre passe-bas de type Butterworth et de fréquence de coupure 2 kHz à l'aide du code suivant : `[B,A] = butter(2,2*2000/100000);`. Vous pouvez visualiser le module de la réponse fréquentielle de ce filtre à l'aide du code suivant :

```
[hcont,f]=freqz(B,A,1024,100000);
plot(f,abs(hcont));grid on;
```

— Appliquer ce filtre au signal de sortie du bloqueur. La sortie du filtre simule alors le signal continu restitué.
(`filter`)

— Interpréter les résultats obtenus en comparant les divers signaux (original, bloqué, restitué).

4 Quand le bruit devient utile

On se propose ici d'étudier sur un exemple un cas où le bruit permet finalement une meilleure quantification du signal : le « dithering ». Cette méthode est très souvent employée dans le traitement des signaux acoustiques, quand les moyens disponibles imposent un codage sur un nombre de bits relativement faible. Par exemple pour le CD : le signal quantifié codé sur 32 bits par exemple, ne sera finalement stocké sur le CD qu'avec une résolution de 16 bits.
Pour des raisons de simplicité, l'opération d'échantillonnage est ici négligée, on travaillera donc directement sur les signaux « continus ».

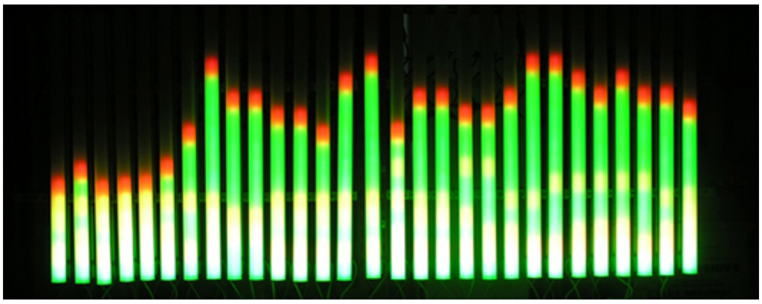
— Représentez le signal original, et ses valeurs quantifiées pour $q = 2$ sur un même graphique.

— Additionner au signal original un bruit blanc de fonction de densité de probabilité uniforme sur l'intervalle $\left[-\frac{q}{2}, \frac{q}{2}\right]$ puis quantifiez ce signal bruité (c'est l'opération de dithering). Représentez le signal quantifié obtenu sur le graphique précédent et notez l'effet du bruit additif sur le résultat.

— Filtrer les deux signaux quantifiés précédents à l'aide du filtre passe-bas servant à la restitution. Comparer visuellement les résultats obtenus avec et sans dithering. Quel signal vous paraît être plus le proche de l'original ? Pour une comparaison plus précise, on pourra également calculer les variances d'erreur de restitution obtenues avec et sans dithering.

Transformée de Fourier Discrète

environ 2 séances



L'objet de ce BE est de mettre en évidence quelques propriétés de la transformée de Fourier discrète (TFD), susceptibles de conduire à des difficultés d'interprétation.

1 Propriétés élémentaires de la TFD

1.1 Synthèse de signaux déterministes

— Synthétiser $N = 256$ points d'un signal (s_1) cosinusoïdal de fréquence $f_1 = 10$ Hz, d'amplitude 1 (Volt par exemple) échantillonné à une fréquence f_e choisie pour avoir exactement 16 périodes dans la fenêtre d'observation de 256 points. La phase initiale (phase en $t = 0$) sera choisie arbitrairement. Visualiser cette série en précisant les valeurs en temps de l'axe des abscisses.

— Synthétiser un signal sinusoïdal s_2 échantillonné à la valeur f_e précédente, mais de fréquence f_2 telle que 15,5 périodes soient observées dans la fenêtre d'observation.

1.2 Fonction `TFDVisu.m`

Afin de pouvoir conduire des analyses élémentaires des propriétés de la TFD, vous allez construire une fonction que l'on appliquera aux différents signaux de test.

— Dans un fichier que l'on nommera `TFDVisu.m`, codez une fonction avec les propriétés suivantes :
  . les arguments d'entrée de la fonction seront le signal à étudier et sa fréquence d'échantillonnage f_e
  . la fonction calculera la transformée de Fourier discrète du signal fourni en entrée par l'algorithme de transformée de Fourier rapide (FFT pour Fast Fourier Transform) (`fft`)
  . la fonction retournera en arguments de sortie les valeurs du module et de la phase de la TFD du signal étudié, ainsi que les fréquences physiques auxquelles elles ont été calculées

— Faites attention aux 2 points suivants :
  . normaliser le module de la TFD pour retrouver les unités physiques du signal
  . les fréquences doivent couvrir la période spectrale principale (c'est à dire de $-\frac{f_e}{2}$ à $\frac{f_e}{2}$)

1.3 Analyse des signaux tests

- Analysez le signal s_1 à l'aide de la fonction `TFDVisu.m`.
  - . Afficher sur une **même figure** le module et la phase de la TFD du signal en fonction de la fréquence physique.
  - . Discuter des valeurs numériques obtenues pour les valeurs du module et de la phase de la TFD en $f = \pm 10\text{Hz}$.
  - . Changer la phase initiale de s_1 et reprendre cette question pour quelques valeurs de phase initiale. Commentaire ?
- Analyser le signal s_2 à l'aide de la fonction `TFDVisu.m`.
  - . Reprendre les questions précédentes pour ce signal et comparer les résultats obtenus.
- Pour un des signaux précédents, analyser (à l'aide de la fonction `TFDVisu.m`) l'influence d'un retard k circulaire ou non sur les valeurs de la TFD.
  (`circshift`)

2 Interpolation par « zero-padding »

2.1 Interpolation de la TFD

Le zero-padding (littéralement « bourrage de zéros ») est une technique simple d'interpolation d'une série régulièrement échantillonnée, très largement répandue, et utilisée dans de nombreux logiciels.

Principe :
Ajouter aux observations (série de longueur N) une série formée de M zéros avant de calculer sa TFD. Le plus souvent, on fixe $P = N + M = 2^k$, ce qui permet de recourir à des algorithmes de calcul rapide de la TFD (comme la FFT). Le spectre du signal est alors inchangé, mais ses valeurs seront calculées pour les fréquences multiples de $\frac{f_e}{P} = \frac{f_e}{2^k}$ et non aux seules valeurs multiples de $\frac{f_e}{N}$. Cette opération réalise donc une interpolation en fréquence, accessible dans le logiciel Matlab en appelant la fonction `fft` avec deux arguments d'entrée : `fft(s,P)` au lieu de `fft(s)`.

- Augmenter la résolution fréquentielle (d'un facteur 16 par exemple) des représentations des TFD de s_1 et s_2, en comparant les abscisses calculées dans chacune des deux analyses (avec $N = 256$, puis $P = 16 \times 256$).
- Interpréter alors l'existence des nombreuses valeurs non nulles dans la représentation spectrale sur P points de s_1 et s_2.
- Avantages et inconvénients de l'interpolation de la TFD ?

2.2 Interpolation d'une série temporelle

Par un raisonnement analogue, on se propose de calculer les valeurs interpolées dans le domaine temporel d'un signal quelconque. Une méthode de zero-padding de la TFD d'une série temporelle pour réaliser une interpolation consiste à ajouter des zéros au spectre de Fourier discret, à des fréquences vérifiant $|f| > \frac{f_e}{2}$. La fonction `ZeroPad.m` fournie avec l'énoncé réalise cette opération.
(`zeros, fftshift, ifft`)

- Expliquer chacune des 4 lignes de code qui constituent `ZeroPad.m`.
- Appliquer cette méthode à l'interpolation de s_1 ou s_2, en prenant soin de bien différencier les points du signal original de ceux obtenus par interpolation.

3 Apodisation, fenêtres de pondération

Les techniques d'apodisation consistent à appliquer des coefficients de pondération sur la fenêtre d'observation du signal, de telle sorte que les oscillations (ou les « pieds ») de la TFD de la fenêtre soient atténuées. Cependant, cela ne peut s'obtenir qu'au prix d'un élargissement du lobe principal de la TFD de la fenêtre. Ces techniques sont extrêmement importantes dans les problèmes de conception de filtres numériques, mais aussi dans les problèmes d'analyse spectrale en général [1].

On a vu (ou on verra) en cours différentes fenêtres usuelles. Ces dernières sont déjà programmées dans Matlab.

— Choisir la fenêtre rectangulaire et 2 fonctions d'apodisation parmi celles proposées par l'aide de Matlab (`window`). Les générer sur une longueur de 32 échantillons.
— Afin de visualiser l'effet de l'apodisation sur la TFD, représenter dans deux sous-fenêtres d'une même figure la fonction d'apodisation en temps ainsi que le module de sa TFD (en dB) avec une résolution fréquentielle augmentée d'un facteur $r = 16$. Attention à la normalisation des ordonnées en fréquence. **Relever les principales caractéristiques de ces fenêtres.**
— Appliquer une méthode d'apodisation à l'analyse du spectre (TFD) de s_1 et s_2 introduits au début de ce BE et interpréter les résultats obtenus. D'après vous, que perd-on et que gagne-t-on en introduisant ces fenêtres ?

4 À vous de jouer ...

Le but est ici d'employer les techniques précédentes pour réaliser l'analyse spectrale d'un signal inconnu et pour en tirer toutes les informations possibles sur son contenu. Charger le signal contenu dans le fichier `SigTest.mat` (`load`). Déterminez du mieux possible les composantes constituant ce signal à l'aide d'une analyse spectrale par TFD.

1. De façon plus surprenante peut-être, ces techniques ont connu récemment un regain d'intérêt en optique ; on établit en effet que l'apodisation (ou application d'une fenêtre de pondération) sur la pupille d'entrée d'un instrument d'optique permet de limiter les effets de diffraction, cruciaux dans les applications de détection d'exoplanètes par exemple.

Synthése de filtres numériques

environ 2 séances

1 Analyse de filtres numériques

On considère les filtres numériques ayant les fonctions de transfert suivantes :

$$G_1(z) = \frac{z^2}{z^2 - 0.4z + 0.60} \tag{1}$$

$$G_2(z) = \frac{z^2 + 1}{z^2 + 0.81} \tag{2}$$

$$G_3(z) = 1 + z^{-1} + z^{-2} \tag{3}$$

- Réalisez tout d'abord l'analyse théorique de ces trois filtres :
  - tracer pour chaque filtre la position des pôles et des zéros dans le plan complexe
  - en déduire l'allure du module de la réponse fréquentielle (méthode géométrique)
  - calculer numériquement (en utilisant $z = e^{j2\pi\lambda}$) cette réponse fréquentielle
- Créer ensuite une fonction (par exemple `FiltreVisu.m`) qui permette de reproduire cette étude théorique de façon automatique :
  - les paramètres d'entrée seront les coefficients du numérateur et du dénominateur, le nombre de fréquences pour lesquelles le module et la phase de la réponse fréquentielle seront calculés, ainsi que tous les autres paramètres nécessaires
  - elle affichera le module et la phase de la réponse fréquentielle du filtre en fonction de la fréquence ainsi que la position des pôles et des zéros du filtre dans le plan complexe
  - les paramètres de sortie seront le module et la phase de la réponse fréquentielle, et le vecteur des fréquences correspondantes

  (`zplane`, `freqz`)

2 Synthèse élémentaire d'un filtre RIF : méthode des fenêtres

On souhaite synthétiser un filtre numérique à réponse impulsionnelle finie (RIF) :
- à phase linéaire de type I
- de type passe-bas
- de fréquence de coupure égale à $f_o = 150$Hz
- pour des signaux échantillonnés à $f_e = 1280$Hz

La méthode mise en œuvre dans cette section sera la méthode des fenêtres.

2.1 Synthèse élémentaire

- Calculer analytiquement la réponse impulsionnelle $h_{id}[n]$ d'un filtre numérique passe-bas idéal de fréquence de coupure f_o pour des signaux échantillonnés à f_e. Représenter sous Matlab cette réponse idéale pour les valeurs de f_o et f_e précédentes.
- Représenter la réponse impulsionnelle des filtres **causaux** $h_1[n]$ et $h_2[n]$ (de taille respective $N_1 = 9$ et $N_2 = 51$) construits par fenêtrage (fenêtre rectangulaire) de la réponse impulsionnelle $h_{id}[n]$.
- Comparer la réponse en fréquence de ces deux filtres.

2.2 Influence du choix des fenêtres et calcul de l'ordre

- Dans le cas $N_2 = 51$, représenter la réponse impulsionnelle et le module de la réponse en fréquence obtenus par la méthode des fenêtres en utilisant les fenêtres de Hann puis de Blackman par exemple.
- Avec le filtre précédent et une fenêtre de Hann, calculer à partir de l'annexe donnée en fin de sujet la valeur minimale de N (impaire) pour obtenir une bande de transition $\Delta\lambda = 0.02$.

2.3 Transposition de filtres

- Proposer une méthode simple permettant de transformer le filtre RIF passe-bas précédent en un filtre RIF passe-haut.
- Réaliser cette transformation et représenter les réponses impulsionnelle et fréquentielle correspondantes afin de vérifier votre démarche.

3 Synthèse de filtres RIF : comparaison de méthodes

Il ne s'agit pas ici de reprogrammer les algorithmes de synthèse vus en cours, ceci dépasse le cadre de ces BEs. Dans cette partie, on s'appuiera sur les fonctions disponibles par défaut dans Matlab.
Le cahier des charges pour le filtre à synthétiser est le suivant :
- filtre passe-bande
- fréquence de coupure basse $f_{o_1} = 150$Hz
- fréquence de coupure haute $f_{o_2} = 300$Hz
- bandes de transition de $\Delta f = 20$Hz
- ondulations maximales en bande passante et coupées égales à 1%
- fréquence d'échantillonnage $f_e = 1280$Hz

3.1 Méthode des fenêtres

- Synthétiser le filtre précédent par la méthode des fenêtres (`fir1`).
- Analyser le filtre ainsi obtenu avec votre fonction `FiltreVisu.m`.

3.2 Méthode minimax

- Synthétiser le filtre « equiripple » par la méthode minimax (algorithme de Parks-McClellan avec `firpmord,firpm`).
- Analyser le filtre ainsi obtenu avec votre fonction `FiltreVisu.m`.

3.3 Méthode des moindres carrés

- Synthétiser le filtre par la méthode des moindres carrés (`firls`).
- Analyser le filtre ainsi obtenu avec votre fonction `FiltreVisu.m`.

3.4 Méthode de la fenêtre de Kaiser

On utilise ici la fenêtre de Kaiser dans la méthode des fenêtres. En effet, cette fenêtre particulière permet de fixer séparément les tolérances en bande passante ou coupée (les « déviations ») et la bande de transition, ce qui constitue une amélioration importante.

- Synthétiser le filtre passe-bas précédent par la méthode des fenêtres en utilisant la fenêtre de Kaiser (`kaiserord, fir1`).
- Analyser le filtre ainsi obtenu avec votre fonction `FiltreVisu.m`.

3.5 Comparaison

- Représenter sur un même graphique les réponses en fréquence obtenues par chacune de ces méthodes pour pouvoir les comparer. On s'intéressera également à la complexité des filtres obtenus.

4 Synthèse de filtres RII

4.1 Synthèse d'un filtre passe-bas : méthode de la transformée bilinéaire

On souhaite concevoir un filtre numérique $H_{1_Z}(z)$ à réponse impulsionnelle infinie (RII) dont le cahier des charges est le suivant :

- filtre passe-bas du second ordre
- de fréquence de coupure à -3dB égale à $f_o = 150$Hz
- de fréquence d'échantillonnage $f_e = 1280$Hz

La méthode utilisée sera celle de la transformée bilinéaire.

- Déterminer la fonction de transfert $H_1(p)$ du filtre de Butterworth analogique satisfaisant le gabarit du cahier des charges précédent. On rappelle qu'un filtre de Butterworth du deuxième ordre de pulsation de coupure à -3dB **unitaire** est décrit par la fonction de transfert

$$H(p) = \frac{1}{p^2 + \sqrt{2}p + 1}.$$

- Tracer le diagramme pôles-zéros (`roots`) et la réponse en fréquence (`freqs`) de $H_1(p)$ et vérifier que le cahier des charges est bien respecté.
- Déterminer la fonction de transfert $H_{1_Z}(z)$ correspondante par la transformation bilinéaire. Prendre en compte le phénomène de distorsion fréquentielle induit par la transformation bilinéaire.
- Tracer le diagramme reliant la fréquence numérique f_n à la fréquence analogique correspondante f_a pour une fréquence d'échantillonnage $f_e = 1280$Hz.
- Tracer le diagramme pôles-zéros et la réponse en fréquence de $H_{1_Z}(z)$ et vérifier si le cahier des charges est bien respecté.
- Représenter sur une même figure les réponses en fréquence de $H_1(p)$ et $H_{1_Z}(z)$ pour des fréquences comprises entre 0 et $2f_e$. Conclure.

4.2 Synthèse d'un filtre passe-bande : méthode de la transformée bilinéaire

On souhaite concevoir, par la transformée bilinéaire, un filtre passe-bande à réponse impulsionnelle infinie dont le cahier des charges est le suivant :

- filtre passe-bande du second ordre
- de fréquences de coupure à -3dB égales à $f_{o_1} = 150$Hz et $f_{o_2} = 300$Hz
- de fréquence d'échantillonnage $f_e = 1280$ kHz
- Spécifier le gabarit que doit vérifier le filtre analogique modèle, puis déterminer le filtre de Butterworth $H_2(p)$ passe-bande satisfaisant ce gabarit. On rappelle qu'un filtre passe-bande peut être

obtenu à partir d'un filtre passe-bas unitaire en appliquant à la fonction de transfert de ce dernier la transformation :

$$p \leftarrow \frac{p^2 + w_{o_1} w_{o_2}}{p(w_{o_2} - w_{o_1})}$$

où w_{o_1} et w_{o_2} sont les pulsations de coupure basse et haute désirées.

— Calculer le filtre numérique $H_{2_Z}(z)$ correspondant par la méthode de la transformée bilinéaire.

— Tracer les réponses en fréquence du filtre analogique $H_2(p)$ et du filtre numérique $H_{2_Z}(z)$ ainsi synthétisés et les comparer.

5 Synthèse de filtres numériques : comparaison

On souhaite concevoir un filtre passe-haut vérifiant les spécifications suivantes :

— fréquence de coupure de la bande coupée $f_c = 100$ Hz

— fréquence de coupure de la bande passante $f_p = 130$ Hz

— ondulations en bande coupée et passante $\delta_c = \delta_p = 1$ %

— fréquence d'échantillonnage $f_e = 1280$ Hz

Utiliser les fonctions intégrées Matlab pour concevoir ce filtre sous une forme :

1. RIF par la méthode des fenêtres (`fir1`) puis la méthode minimax (`firpmord, firpm`),
2. RII par la méthode de la transformée bilinéaire en employant un prototype analogique de Butterworth (`buttord, butter`) puis elliptique (`ellipord, ellip`).

Pour les filtres obtenus, comparer :

— leur réponse fréquentielle en module et en phase,

— l'emplacement de leurs pôles et leurs zéros,

— leur complexité respective.

Conclure sur leurs avantages et inconvénients respectifs.

A Annexe

Caractéristiques de quelques fenêtres de pondération.

Window	$\Delta\lambda$	Passband ripple	Main/Side lobe (dB)	Stopband attenuation (dB)	function
Rectangular	$\frac{0.9}{N}$	0.7416	13	21	1
Hann	$\frac{3.1}{N}$	0.0546	31	44	$0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right)$
Hamming	$\frac{3.3}{N}$	0.0194	43	53	$0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right)$
Blackman	$\frac{5.5}{N}$	0.0017	57	74	$0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$

Données expérimentales

Analyse d'une courbe de charge d'un réseau électrique

environ 2 séances

Durant ce BE, vous allez pouvoir employer sur un signal expérimental les techniques et les outils de traitement numérique du signal que vous avez découvert lors des BE précédents. L'objectif est d'analyser puis de traiter la consommation électrique instantanée de la France métropolitaine enregistrée sur une durée de plusieurs années.
Pour réaliser **l'analyse de cette série temporelle**, vous devrez réaliser son **analyse spectrale** afin d'en déterminer ses principales composantes.
Pour réaliser **le traitement de ces données**, vous devrez concevoir et employer des **filtres numériques** afin de séparer les composantes identifiées.

1 Contexte et grandeur à analyser

Les gestionnaires de réseaux électriques tels que RTE cherchent constamment à atteindre l'équilibre entre production et consommation d'électricité afin de maintenir la stabilité du réseau. Cet équilibre ne peut être approché qu'en prédisant correctement la consommation électrique afin d'organiser la production en conséquence. Or les méthodes de prédiction d'une telle grandeur s'appuient principalement sur son caractère cyclique. Il est donc fondamental de connaître précisément la fréquence et l'amplitude de chacun des cycles constituant la consommation électrique.
Une étude des habitudes des consommateurs d'électricité conduit très vite à l'identification des trois principaux cycles temporels constituant cette série :

- un cycle annuel, comportant un pic de consommation en hiver, un creux en été, ainsi que divers évènements plus localisés (jours fériés, vacances, évènements importants...)
- un cycle hebdomadaire, où apparaissent clairement les 5 jours ouvrables avec une consommation globalement forte et le week-end où la consommation diminue
- un cycle quotidien, constitué du creux de nuit correspondant au minimum de consommation sur les 24 heures, du pic du matin, du creux d'après-midi et du pic du soir. On peut noter que le maximum de consommation est atteint lors du pic du matin en été, et du pic du soir en hiver

Les données que vous devrez traiter sont publiques et disponibles sur le site opendata.reseaux-energies.fr.

- Téléchargez les données de consommation quotidienne brute d'électricité en France métropolitaine sur une période allant au moins du 01/01/2016 au 31/12/2018. En cas de problème, rabattez-vous sur le fichier `conso_elec_20160101_20181231.csv` fourni avec le sujet de BE.
- Visualiser ce signal et retrouvez des portions représentatives des cycles présentés plus haut.

2 Analyse spectrale

2.1 Préliminaires

Le but est ici d'élaborer un programme permettant de réaliser simplement l'analyse spectrale d'une série expérimentale.

- Basez vous sur la fonction `TFDVisu()` que vous avez développée lors du BE sur la transformée de Fourier discrète pour élaborer une fonction `spectre()` permettant de d'obtenir le spectre du signal :
  - on ne s'intéresse ici qu'au spectre d'amplitude
  - ne représenter que la partie du spectre correspondant aux fréquences positives
  - normaliser le résultat pour obtenir en ordonnée l'amplitude de chaque sinus composant la série
- Testez et validez votre fonction sur un signal synthétique de votre choix constitué par exemple de quelques sinus.

2.2 Analyse spectrale

L'objectif est ici de déterminer les composantes et les plages fréquentielles correspondant à chacun des cycles précédents.

- À l'aide de votre fonction `spectre()`, réalisez une analyse spectrale du signal de consommation électrique et représentez son spectre d'amplitude.
- Exprimez la fréquence d'échantillonnage f_e dans une unité adaptée aux variations du signal analysé afin d'obtenir une échelle facilement lisible pour l'axe des fréquences. En effet, une fréquence f est par définition l'inverse d'une période temporelle T. Cette dernière se mesurant souvent en secondes, f s'exprime donc souvent en « cycles par seconde » ou Hertz. Si l'échelle de la seconde n'est pas adaptée pour représenter la période T, vous pouvez employer une autre échelle de temps comme par exemple le siècle. Le contenu fréquentiel d'une telle grandeur s'exprimera alors en « cycles par siècle » et non plus en Hertz.
- Exprimer ce spectre en échelle linéaire et en déciBels. Commentez le résultat observé.
- Pour finaliser votre analyse, identifiez dans le spectre obtenu les composantes et les bandes de fréquence correspondant à chacun des cycles constituant ce signal (annuel, hebdomadaire, quotidien).

3 Filtrage numérique

L'objectif est maintenant de séparer, par filtrage RIF, les différents cycles composant cette grandeur.

- À partir des résultats de l'analyse spectrale que vous venez de réaliser sur ce signal, déterminez les gabarits des filtres permettant de séparer les composantes correspondant à chacun des cycles ci-dessus.
- Synthétisez les filtres RIF correspondants, puis appliquez les au signal à traiter.
- Commentez la qualité de vos résultats en comparant l'allure temporelle et le spectre des signaux avant et après filtrage.