TSA ATIAM, TP1 : RAPPELS MATLAB ET QUELQUES EXEMPLES LOUPE SPECTRALE

Bertrand David & Roland Badeau, Octobre 2012

MATLAB est un environnement interactif adapté au traitement numérique du signal comportant du calcul matriciel. Il permet de travailler interactivement en passant des commandes au clavier, ou de réaliser des programmes en plaçant ces commandes dans des fichiers de texte (scripts, avec l'extension .m). Ceux-ci sont exécutables, le nom du fichier devenant une commande.

1 Prise en main

Lancement du logiciel sur PC Le logiciel se lance depuis l'icône situé sur le bureau. Une invite de commande ("prompt") s'affiche dans la fenêtre principale dite *de commande* (celle de droite). A l'aide de la commande **`cd`** vous fixerez votre répertoire de travail à l'emplacement de d :\user\ où seront rangés vos fichiers personnels.

Une session MATLAB consiste en une suite de commandes lancées au clavier et exécutées immédiatement par MATLAB. MATLAB maintient une pile des commandes passées. On peut les rappeler et les éditer à l'aide des touches flèches $\rightarrow \leftarrow \uparrow \downarrow$. On quitte l'application avec la commande `quit`, ou en refermant la fenêtre Windows.

Les commandes et les variables MATLAB ne connaît qu'un seul type de variable : les matrices de complexes. Un scalaire (réel ou complexe) est traité comme une matrice 1x1. Un vecteur est une matrice 1xN ou Nx1 (MATLAB distingue les vecteurs lignes et les vecteurs colonnes). Tous les nombres sont codés en virgule flottante double précision (8 octets). Pour créer une matrice 2x2, on pourra par exemple taper :

```
x = [1 2; pi 0]
x = [1 2 , pi 0] % résultat ≠ de la ligne de commande précédente
```

Les chaînes de caractères sont aussi des vecteurs, chaque code ASCII étant codé comme un réel en double précision. Elles sont écrites entre apostrophes simples :

```
nom = 'coco'
```

et peuvent être manipulées comme des vecteurs numériques. Essayer :

```
nom(1) = nom(1) +1
```

Comme en langage C, MATLAB distingue les majuscules des minuscules dans les noms de variables. La virgule ou le point-virgule permettent de séparer plusieurs commandes sur la même ligne. Le résultat de chaque commande est affiché à l'écran, sauf si la commande est terminée par un ";". Son emploi est recommandé à la fin de chaque instruction pour éviter les affichages inutiles (lorsqu'ils sont longs, ils peuvent être interrompus en tapant Ctrl-C).

L'espace de travail L'espace de travail est constitué de l'ensemble des variables allouées en mémoire vive. MATLAB est un langage semi-interprété, ce qui fait que, contrairement au langage C, les variables n'ont pas besoin d'être déclarées au préalable. MATLAB les crée (ou adapte leurs dimensions) au fur et à mesure de l'exécution des commandes. Les commandes **`who`** et **`whos`** permettent de connaître les variables existantes et leurs dimensions. La commande `clear` détruit toutes les variables et libère l'espace de travail.

Les programmes Ce sont des fichiers de texte, dont le nom se termine par l'extension .m. On les crée en utilisant la commande du menu Edit/New/MFile. Ils sont de deux types :
– les scripts, qui contiennent des suites de commandes telles que l'on pourrait les entrer au clavier,
– les fonctions, qui sont l'analogue des fonctions en C.
Pour exécuter un script, il suffit de taper le nom du fichier au clavier (sans l'extension .m). Une fonction doit débuter par une ligne de définition, et peut comporter des variables de sortie qui doivent être définies dans le corps de la fonction. Par exemple :

```
function [mod, ph] = mafonc(sig, Nfft)
tf = fft(sig, Nfft); % calcule la TFD de sig sur Nfft points
mod = abs(tf); % calcule son module
ph = angle(tf); % calcule sa phase
```

Le nom du fichier DOIT correspondre au nom de la fonction (ici mafonc.m). Pour utiliser la fonction, il suffit de passer les commandes suivantes :

```
sig = [1 1 1];
[M, P] = mafonc(sig, 512);
plot(M);
```

Exécuter un script est équivalent à taper les commandes au clavier. En particulier toutes les variables crées par le script subsistent dans l'espace de travail une fois le script terminé. Au contraire, les variables créées par une fonction sont locales et ne sont pas conservées une fois la fonction terminée (sauf bien sûr les variables de sortie). Il est toujours préférable de travailler avec des scripts plutôt que directement au clavier. La correction des erreurs est facilitée et on conserve une trace du travail effectué.

L'aide en ligne MATLAB comporte deux types d'aide :

– une aide WINDOWS, accessible par le menu Help de la fenêtre principale,
– une aide en ligne, à l'aide de la commande help. Par exemple, pour obtenir de l'aide sur la fonction `fft`, taper : `help fft`. Cette aide concerne toutes les fonctions, y compris celles qui ont été écrites localement (comme par exemple la fonction `mafonc` ci-dessus). Pour qu'une fonction dispose d'une telle aide en ligne, il suffit d'écrire le texte d'aide dans des lignes de commentaires (débutant par %) en dessous de la ligne de déclaration "`function` ..." (Le texte affiché par help... s'arrête à la première ligne de commande ou à la première ligne vide).

ATTENTION : dans l'aide WINDOWS, les constantes sont données en majuscules (par ex. PI) alors qu'elles sont en fait en minuscules (pi).

2 premier.m

Les morceaux de code matlab fournis sont donnés à titre d'exemples dont vous pourrez vous inspirer pour réaliser votre script.

Il s'agit ici de programmer un filtre RIF comme un produit scalaire par bloc. Vous créerez un *script* premier.m, qui comportera les étapes suivantes :
– définition et initialisation des variables `x` (signal) et `h` (RI du filtre) comme des vecteurs lignes. ex :
```
x = 1:10; % cree un vecteur ligne qui contient 1,2,...,10
h = [1 -1 0 2]'; % cree un vecteur ligne et le transpose : h est 4x1
h = ones(4,1); % vecteur colonne composé uniquement de 1
```
– création et initialisation à zéro du vecteur ligne de sortie. Que vaut la longueur du produit de convolution $x \star h$ si les longueurs respectives de x et h sont N et M ? Une solution pour traiter le transitoire du filtrage RIF consiste à rallonger le vecteur d'entrée du bon nombre de zeros.
Fonctions utiles :
```
zeros(P,Q); % cree une matrice PxQ remplie de zeros
sortie = zeros(1,10); % vecteur ligne de 10 échantillons
length(x); % renvoie la longueur du vecteur x
```

```
[L,C]= size(x); % renvoie le nbre de lignes et de colonnes de x
```

– boucle sur les échantillons de sortie. Syntaxe :
```
for k = 1:length(x)
....
end
```
Il existe de même les commandes '`if`' et '`while`'.
– dans le corps de la boucle, calcul du produit scalaire. Il s'agit simplement du produit d'un vecteur ligne par un vecteur colonne. ex :
```
x = 0:pi/3:pi;
y = ones(1,4);
z = x * y';
```
– on mesurera le temps de calcul à l'aide des fonctions : `tic` ; `toc` ; `etime(t, clock)` ; `% cf help`

Rques :
– les opérations courantes `+ , - , * , / , ^ ,` sont définies matriciellement. Pour réaliser les opérations terme à terme, il faut faire précéder d'un ".". ex : `[1 2 3].*[1 2 3] = [1 4 9]`. Une exception toutefois : l'opération avec un scalaire : `[1 2 3]*3 = [3 6 9]` et `[1 2 3]+3 = [4 5 6]`.
– le ' sert à transposer et à conjuguer. Attention quand les grandeurs sont complexes ! ex : essayer : `x = [1+i -2*i 3]; x' , x.'`.
– la multiplication d'un vecteur colonne par un vecteur ligne crée une matrice. Essayer : `[1 ; 2; 3]*[1 2 3]`.

Application : on crée une rampe bruitée de 100 points par

```
N = 100;
n = 0:N-1;
x = n + randn(1,N);
```

utiliser un rif moyenneur ($h(n) = 1/M$ pour $n = 0...M-1$) pour lisser cette rampe et observer l'effet de l'augmentation de l'ordre du filtre. Pour observer on utilisera :

```
figure; % ouvre une nouvelle fenêtre graphique
plot(x); % affiche x
plot(n,x,'r',n,y,'y'); % affiche x (en rouge) et y (en jaune) en
% fonction de n.
```

3 Affichage d'un sinus - Stroboscopie

On cherche à observer les effets sur l'affichage d'un sinus lorsqu'on fait varier sa fréquence réduite. Le début du script est fourni :

```
% affichage d'un sinus de fréq. réduite ν0
N = 1000; % nombre de points du sinus
nu0 = .1; % fréquence réduite du sinus
% construction du signal
n = 0:N-1;
s = sin(2*pi*nu0*n);
```

c'est à vous de programmer la partie affichage :
– affichage de tout le signal
– demande interactive des valeurs à afficher (entre `nMin` et `nMax`) sous la forme : `nMin = input('borne inférieure =?');`
– affichage de cette partie seulement. Deux choix de programmation : soit en jouant sur les indices par l'affichage de `st = s(nMin:nMax);` soit en utilisant la commande `axis([...]) % cf help axis`

Observer les effets optiques de la variation de $\nu 0$ (essayer par exemple $\nu 0 = 0.49$). Interpreter ces effets en considérant d'une part l'aspect temporel de l'échantillonnage d'une sinusoïde et d'autre part l'aspect fréquentiel. On remarquera que le comportement graphique de MATLAB pour le tracé des courbes est équivalent à celui d'un bloqueur d'ordre 1.

Rque : autres commandes graphiques utiles : `subplot , hold , xlabel , ylabel , title, figure,`

`stem.`

4 Observation du spectre par la TFD

On cherche ici à observer :

1. les effets de l'échantillonnage de la TF lorsqu'on utilise la TFD
2. les effets du fenêtrage temporel sur le spectre observé à l'aide d'une TFD. L'ordre de la TFD sera alors pris suffisamment grand pour éviter les problèmes liés à l'échantillonnage du spectre.

Spécifications : créer une *fonction* Matlab de la forme :

```
obspec(nu0,N,Nfft,fen); % fonction sans paramètre de sortie
```

qui affiche le spectre d'un cosinus de fréquence réduite ν_0, et de longueur N fenêtré par la fenêtre `fen` de même longueur. On pourra utiliser les fonctions :

```
w = hanning(15); % crée une fenêtre de hanning sous forme de vecteur colonne
% autres exemples : hamming, blackman
Nfft = 2^(nextpow2(s)+4); % la puissance de deux 16 fois supérieure à
% la longueur du signal
fft(x.*w,Nfft); % calcule la fft de x fenêtré par w sur Nfft points
```

Application :

1. Prendre $\nu_0 = 0.1, N = 64$, et une fenêtre rectangulaire. Essayer les valeurs `Nfft = 32, 64 , 128 et 1024.` Refaire la même étude avec $\nu_0 = 0.125$. Expliquer les observations.
2. Prendre $\nu_0 = 0.1, N = 64$ et $Nfft = 1024$ et comparer le cas des fenêtres rectangulaire et de Hanning. A quel compromis en terme de résolution (ou pouvoir de séparation) doit-on faire face ?

5 Réalisation d'une loupe spectrale

Dans cette partie, vous écrirez un script qui répond au cahier des charges spécifié. L'objectif de ce script est d'effectuer une interpolation spectrale, c'est à dire une dilatation du spectre autour d'une fréquence centrale ν_0 donnée. L'intérêt de cette opération, utilisée par la fonction "zoom" des analyseurs de Fourier, est d'augmenter la résolution d'affichage d'un spectre calculé par transformée de Fourier discrète, sans pour cela devoir utiliser un ordre élevé pour la TFD. Cette application met en évidence les problèmes de repliement de spectre dus à la compression des signaux, illustre certaines propriétés de la TFD, montre l'utilisation de séquences complexes, et met en oeuvre un filtrage discret.

5.1 Principe du traitement

La succession des opérations à effectuer est représentée sur le diagramme présenté figure 1 .

On s'intéresse au cas d'un signal réel dont la puissance est concentrée autour d'une fréquence réduite ν_0. Le signal pris en exemple sera un signal obtenu par modulation de phase.

$x(n) = \cos(2\pi\nu_0 n + \beta \sin(2\pi\nu_m n))$

Les étapes du traitement sont les suivantes :

1. translater le spectre de $x(n)$ en multipliant cette séquence par une séquence $w_1(n)$ que l'on déterminera de manière à ramener la fréquence d'intérêt ν_0 à la fréquence nulle. On obtient $x_1(n)$.
2. filtrage par le filtre passe bas réel de réponse en fréquence $H(e^{j2\pi\nu})$ pour obtenir le signal $x_2(n)$. Le filtre sera réalisé de manière à éliminer le motif non-centré du spectre.
3. décimation du signal $x_2(n)$ d'un facteur M pour obtenir le signal $x_3(n) = x_2(nM)$.

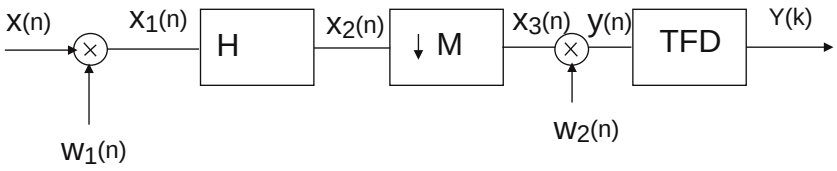

FIGURE 1 – chaîne de traitement pour une loupe spectrale

4. multiplication de $x_3(n)$ par $w_2(n)$ pour centrer le spectre autour de la fréquence réduite 0.5 (séquence $y(n)$).
5. TFD d'ordre N pour observer le spectre obtenu.

5.2 Développement et programmation

On fixe $\nu_0 = 0.3$ (porteuse), $\beta = 10$, $\nu_m = 5.10^{-4}$ et $N = 256$.

Compréhension. Tracer l'allure des spectres (modules de TFD) correspondants à chaque étape décrites précédemment. On remarquera notamment que la décimation est équivalente à un sous échantillonnage du signal analogique. En déduire qu'il existe une valeur maximum de M au delà de laquelle des distorsions apparaissent. De quel type de distorsions s'agit-il ? On appelle $\Delta\nu$ la largeur du support spectral de $x(n)$. Une valeur approchée est fournie par la formule de Carson :

$\Delta\nu \approx 3(\beta + 1)\nu_m$

Calculer la valeur numérique maximale possible de M, avec comme contrainte que M soit une puissance de 2 (algorithmes de FFT).

Zoom rudimentaire Générer la séquence $x(n)$ sur une longueur MN. Observer le spectre à l'aide d'une TFD d'ordre MN et une fenêtre de Hanning. Effectuer un zoom rudimentaire autour de ν_0 en utilisant les fonctionnalités graphiques de MATLAB (icône '+' dans la figure).

Synthèse du filtre. Réaliser l'étape 1 du traitement et observer le spectre obtenu (TFD d'ordre MN, fenêtre de Hanning).

Le filtrage du signal $x_1(n)$ est réalisé par un filtre à RIF dont la synthèse s'effectuera à l'aide de l'algorithme d'échange de Remez, disponible sous forme de fonction MATLAB :

```
d1 = 0.1;
d2 = 1/100; % par exemple
h = firpm(L-1,2*[0 nuC nuA 0.5],[M M 0 0],[d2 d1])
```

L'ordre du filtre (`L-1`) est donné avec une bonne précision par

$$\frac{2}{3(\nu_A - \nu_C)} \log_{10}\left(\frac{1}{10\delta_1\delta_2}\right)$$

Les notations sont celles habituellement utilisées pour spécifier les gabarits de filtres, rappelées figure 2.

Déterminer les valeurs de ν_A et ν_C pour avoir l'ordre du filtre le plus faible (complexité la plus faible).

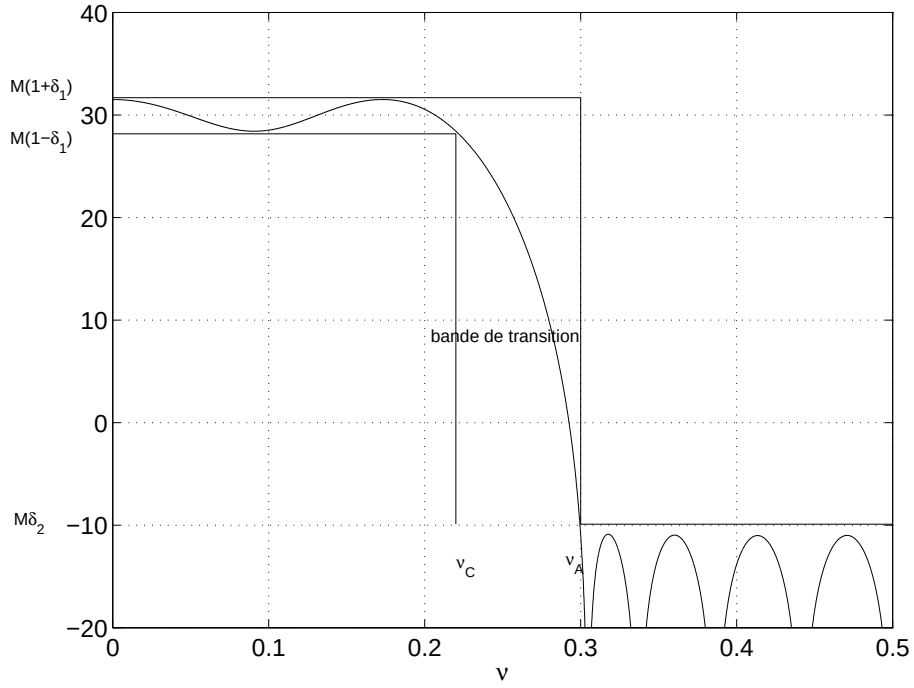

FIGURE 2 – Spécifications d'un gabarit de filtre

Quel est la longueur du transitoire d'un tel filtre ? En déduire que la séquence $x(n)$ de départ doit être de longueur $MN + P$; on précisera la valeur de P.

Le signal $x_2(n)$ est le signal filtré, tronqué de P échantillons. La fonction MATLAB réalisant le filtrage est la fonction

```
x2 = filter(b,a,x1); % x2 est la sortie du filtre
% dont les coefficients de la fonction de transfert sont
% les vecteurs a et b.
```

Observer le spectre de $x_2(n)$ en comparant les résultats obtenus dans le cas d'une fenêtre de Hanning et une fenêtre rectangulaire. Régler les valeurs δ_1 et δ_2 pour obtenir 60dB d'atténuation pour la partie résiduelle du spectre, non-centrée sur 0.

Dernières étapes Pour réaliser la décimation (compression) d'un facteur M on peut utiliser la syntaxe MATLAB suivante :

```
x3 = x2(1:M:M*N);
```

Réaliser la TFD d'ordre N de y et comparer le résultat avec le zoom rudimentaire obtenu précédemment.