2
Applications

Dans ce chapitre, nous allons vous présenter des applications de traitement du signal simples effectuant des opérations de fenêtrage, convolution, modulation-démodulation, filtrage et analyse spectrale. Elles nous permettront d'introduire des fonctions Scilab spécifiques au signal mais aussi de montrer comment construire des boîtes de dialogue pour récupérer des valeurs utilisateurs.

La première application permet d'étudier les effets d'une fonction de fenêtrage choisie par l'utilisateur sur le spectre d'un signal. La deuxième utilise la convolution pour simuler un son dans une autre ambiance que celle où il a été enregistré. Dans notre exemple, nous simulerons le bruit d'un éléphant dans un bâtiment où ces derniers sont interdits ! La troisième application met en jeu la modulation et la démodulation d'amplitude de signaux à l'aide de filtres à réponse impulsionnelle finie ou infinie. Les signaux à moduler sont choisis par l'utilisateur parmi les fichiers audio disponibles sur son disque dur, puis filtrés avec un passe-bas pour respecter la bande passante autorisée par l'onde porteuse. Nous nous intéresserons ensuite à l'intercorrélation entre signaux en créant un programme pour localiser un son au milieu du bruit. Enfin, nous montrerons comment effectuer une analyse spectrale par différentes méthodes, en utilisant le modèle ARMA ou le périodogramme de Welch implémentés dans Scilab ou les méthodes de Capon ou Lagunas du module externe Time Frequency Toolbox. L'utilisateur choisit la source du signal, microphone, simulation ou fichier et peut aggrandir une zone particulière du spectre en cliquant dans la zone graphique pendant l'acquisition, ce qui nous permettra d'introduire un gestionnaire d'événements.

2.1. Fenêtres de pondération

Le traitement du signal numérique porte sur une durée finie d'observation du signal analogique. Le signal analogique est multiplié par une fenêtre temporelle rectangulaire. Ce fenêtrage n'est pas sans conséquence sur le spectre du signal. Un signal dont le spectre est un Dirac est ainsi transformé en un spectre composé d'un lobe principal et de multiples lobes secondaires. La mesure de l'amplitude de deux raies

voisines est alors faussée. Afin d'atténuer ce phénomène, le signal est multiplié par une nouvelle fenêtre dite *de pondération* ou d'*apodisation*.

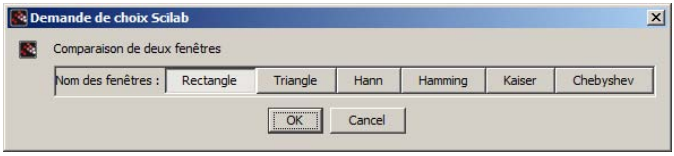
La fonction Scilab `window` permet de construire une fenêtre de pondération de type rectangulaire, triangulaire, Hamming, Hann, Kaiser ou Chebychev. Le programme suivant utilise cette fonction pour analyser un signal contenant deux sinus de même amplitude à deux fréquences f1 et f2. L'analyse est répétée plusieurs fois en laissant la fréquence f1 fixe et en approchant la fréquence f2 de f1. Pour chaque analyse, nous déterminons le nombre de raies mesurées et mesurons la hauteur de chaque raie.

```
// Fréquence d'échantillonnage Fe et N nombre d'échantillons
Fe=11025;
N=2048;
// Amplitudes des fréquences f1 et f2 présentes dans le signal
A1=1;A2=1;
// La fréquence 1 est un multiple de Fe/N
indNu1=58;
nu1=indNu1*Fe/N;
```

Le programme propose à l'utilisateur de choisir le type de fenêtre de pondération à l'aide d'une boîte de dialogue munie de boutons. Celle-ci est créée avec la fonction Scilab `x_choices` prenant deux paramètres et ayant pour résultat le numéro du choix validé par l'utilisateur. Le premier paramètre est le titre de la boîte de dialogue et le second une liste composée de la liste des choix possibles. La structure de cette liste est imposée par `x_choices` : son premier élément correspond au titre des choix possibles, le deuxième au choix par défaut lors de l'affichage de la boîte de dialogue et le suivant à un tableau de chaînes de caractères contenant les choix de fenêtres proposés à l'utilisateur. Pour créer cette liste, nous utilisons la fonction Scilab `list`.

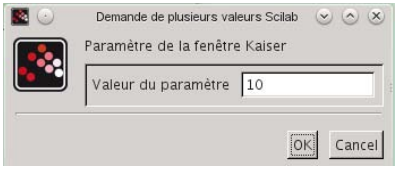
```
// Choix de la fenêtre par l'utilisateur ❶
fenetre=['re';'tr';'hn';'hm';'kr';'ch'];
// Construction d'une liste compatible avec la fonction x_choices
// titre, indice du choix par défaut, puis choix possibles
listeFenetre=list('Nom des fenêtres : ',1,
['Rectangle','Triangle','Hann','Hamming','Kaiser','Chebyshev']);
chxFen=x_choices('Comparaison de deux fenêtres',list(listeFenetre));
// chxfen est l'indice du bouton cliqué par l'utilisateur
// ou -1 si l'utilisateur a choisi Annuler
```

Figure 2.1 : Choix des fenêtres de pondération



Pour les fenêtres de pondération Kaiser et Chebychev, un paramètre numérique supplémentaire requis est demandé à l'utilisateur par une nouvelle boîte de dialogue en utilisant la fonction Scilab `getvalue`. Cette fonction permet de récupérer une ou plusieurs valeurs ayant un type spécifique. Son premier paramètre correspond au titre de la boîte de dialogue, le deuxième à un vecteur de chaînes de caractères composé des noms des champs à récupérer ; le troisième à une liste composée des types et dimensions de chaque valeur saisie. Le dernier, enfin, est une liste de chaînes de caractères des valeurs par défaut affichées dans la boîte de dialogue.

Figure 2.2 : Saisie du paramètre numérique pour la fenêtre de Kaiser.



Note *› Pour plus d'informations sur la création des boîtes de dialogue et la manipulation des listes sous Scilab, vous reporter au module I. Les fondamentaux.*

```
chxFen=x_choices('Comparaison de deux fenêtres',list(listeFenetre));
// Pour les fenêtres Chebyshev et Kaiser un paramètre supplémentaire
// est nécessaire
if  chxFen==5  then
    [ok beta]=getvalue('Paramètre de la fenêtre Kaiser','Valeur du
 paramètre',list('row',1),"10")
end
if  chxFen==6  then
    [ok beta]=getvalue('Paramètre de la fenêtre Chebyshev','Valeur du
 paramètre compris entre 0 et 0.5',list('row',1),"0.2")
end
```

La fenêtre de pondération est ensuite calculée selon le choix de l'utilisateur avec la fonction `window`. Le premier paramètre est une chaîne de caractères composée des deux premières lettres du nom du type de fenêtre choisie, sélectionnée à l'aide de l'indice

chxFen dans la matrice fenêtre créée précédemment (voir ❶). Le second paramètre est la dimension de la fenêtre (le nombre de colonnes). Un troisième paramètre est nécessaire pour les fenêtres de Kaiser et Chebyshev permettant de paramétrer la forme de la fenêtre. La valeur de ce paramètre est égale à la valeur de la variable beta retournée par l'appel précédent à la fonction getvalue.

```
// Construction de la fenêtre de pondération avec N éléments
select(chxFen)
case  5  then
    w=window(fenetre(chxFen),N,beta);
case  6  then
    w=window(fenetre(chxFen),N,[beta -1]);
else
    w=window(fenetre(chxFen),N);
end
```

L'espacement entre deux raies du spectre discret dépend de Fe/N, rapport égal à l'inverse de la durée d'observation du signal. Pour obtenir un tracé du spectre de meilleure résolution, nous augmentons le nombre d'échantillons en utilisant la technique d'ajout de zéros (*zero padding*) au signal temporel.

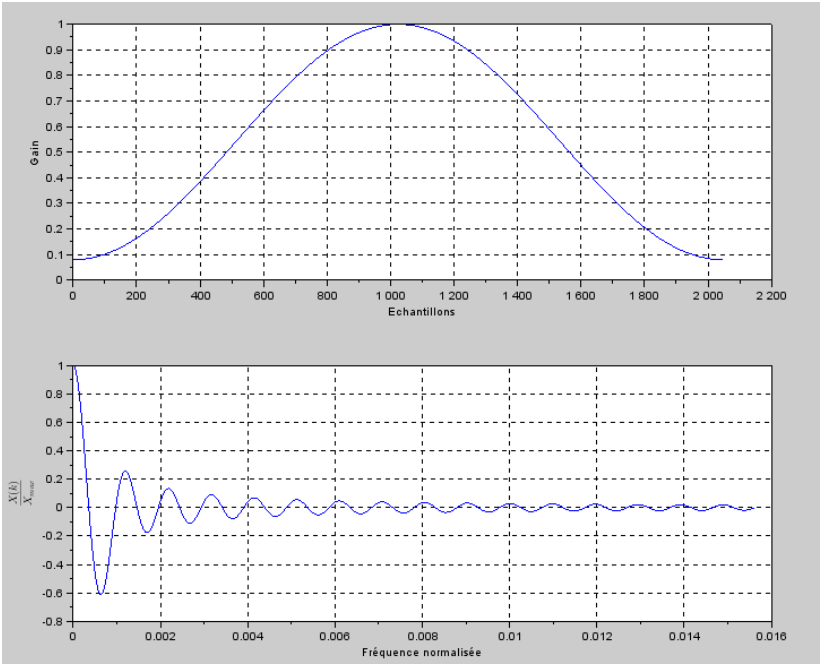
Note *› Pour plus d'informations sur cette technique, voir Bibliographie [R6], chapitre XVI-13.*

```
// Ajout de 0 (63*N zéros) à la fenêtre de pondération
// pour augmenter la résolution spectrale
P=64;
w1=[zeros(1,P*N) ];
// La symétrie de la fenêtre est conservée
w1(1:N/2)=w(1:N/2);
w1(P*N:-1:P*N-N/2+1)=w(N:-1:N/2+1);
```

L'aspect temporel de la fenêtre de pondération est tracé sur le graphe du haut de la fenêtre graphique n° 1 et son spectre sur le graphe du bas (voir Figure 2.3).

```
figure(1);
clf();
subplot(2,1,1);
plot((0:N-1),w)
xlabel('Echantillons');
ylabel('Gain')
xgrid();
subplot(2,1,2);
W=fft(w1);
indice=(0:N)
plot(indice/(N*P),(W(indice+1)/max(abs(W))))
xlabel('Fréquence normalisée')
ylabel('$\frac{X(k)}{X_{max}}$')
xgrid();
```

Figure 2.3 : Aspect temporel et fréquentiel de la fenêtre de pondération



Pour chaque tour de la boucle, nous mesurons la position et l'amplitude des pics associés. La fonction définie dans `DetectionExt` permet de trouver où sont les éléments d'un vecteur supérieurs aux éléments des colonnes adjacentes. Elle utilise la fonction Scilab `diff` qui calcule la différence entre deux éléments consécutifs du vecteur.

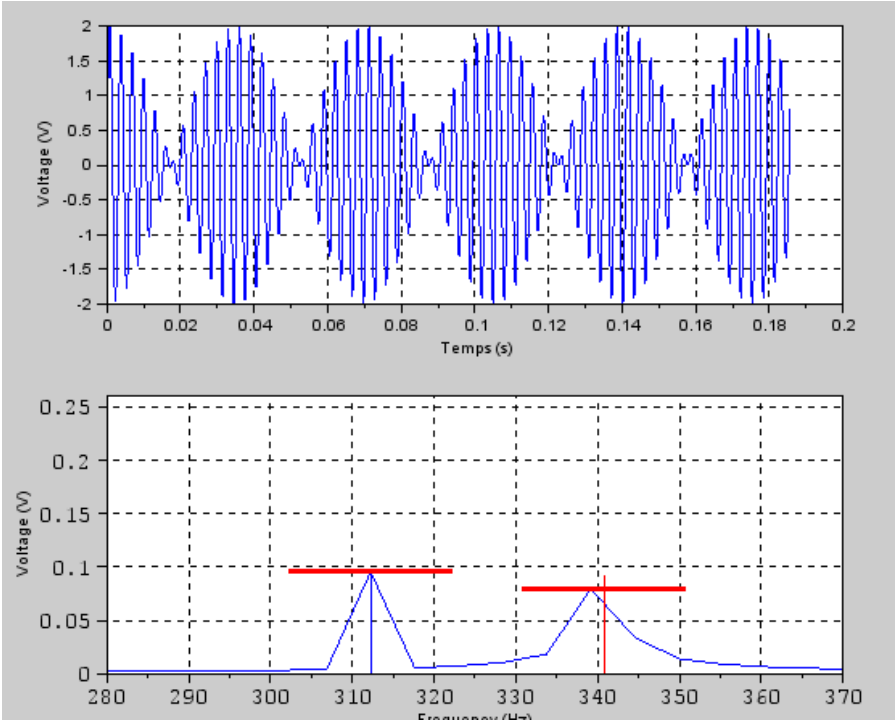
```
function  n=DetectionExt(v)
    // recherche du maximum de v
    vMax=max(v);
    // recherche des valeurs inférieures à 1/3 max
    // et annulation de ces valeurs
    ind=find(v<vMax/3);
    v(ind)=0;
    // calcul de la pente à droite de chaque élément de v
    d=sign(diff(v));
    // Produit des pentes à droite et à gauche des éléments de v
    s=d(2:$).*d(1:$-1);
    // Les valeurs de s négatives sont associées à des maximuns locaux

    // et on le marque à 1
    ind=find(s>0);
    s(ind)=0;
```

```
    ind=find(s<0);
    s(ind)=d(ind);
    n=[0  s  0];
endfunction
```

À chaque tour, le produit du signal par la fenêtre de pondération est tracé sur le graphe du haut de la fenêtre graphique n°2 ; et sur la partie basse, le module du spectre du signal correspondant (voir Figure 2.4).

Figure 2.4 : *Aspect temporel et fréquentiel du produit du signal par la fenêtre de pondération*



Lorsque les deux fréquences se rejoignent, deux valeurs s'affichent sur le graphe du bas : la première est la *résolution*, distance minimale à partir de laquelle les deux raies sont vues dans le spectre ; et la seconde, le rapport maximal mesuré entre les deux raies. Normalement ce rapport devrait être de 0 dB puisque les deux raies ont la même amplitude. La présence de lobes secondaires dans le spectre de la fenêtre de pondération et l'échantillonnage modifient la mesure du rapport des amplitudes des deux fréquences.