- `Wn` bande passante du filtre (fréquence haute de la bande passante pour un passe-bas ; fréquence basse de la bande passante pour un passe-haut ; fréquences basse et haute de la bande passante pour un passe-bande ; fréquences basse et haute de la bande coupée pour un coupe-bande). Les fréquences de `Wn` sont normalisées par rapport à la fréquence de Nyquist.
- `Rp` atténuation maximale (en dB) dans la bande passante.
- `Rs` atténuation minimale (en dB) dans la bande coupée.

3.3.5 Estimation de l'ordre des filtres

Enfin, Matlab, possède des fonctions permettant d'estimer l'ordre minimal nécessaire pour la construction d'un filtre passe-bas ou passe bande entrant dans un gabarit donné :

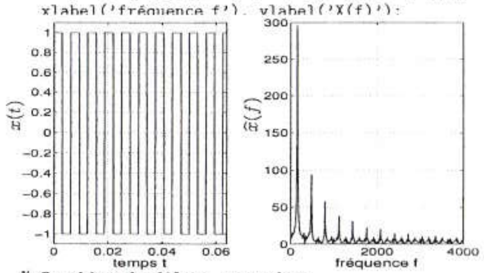
```
>> [n, Wn] = buttord(Wp,Ws,Rp,Rs) ;
>> [n, Wn] = cheb1ord(Wp,Ws,Rp,Rs) ;
>> [n, Wn] = ellipord(Wp,Ws,Rp,Rs) ;
```

- `Wp` bande passante.
- `Ws` bande coupée.
- `Rp` atténuation maximale (en dB) dans la bande passante.
- `Rs` atténuation minimale (en dB) dans la bande coupée.
- `n` ordre du filtre.
- `Wn` fréquence propre du filtre numérique. Pour un filtre passe-bas `Wp` et `Ws` sont les fréquences hautes de la bande passante et basse de la bande coupée. Pour un filtre passe-bande, `Wp` contient les fréquences basse et haute de la bande passante et `Ws` les fréquences haute et basse de la bande coupée.
  Attention, les fréquences sont normalisées par rapport à la fréquence de Nyquist $= \frac{f_e}{2}$.

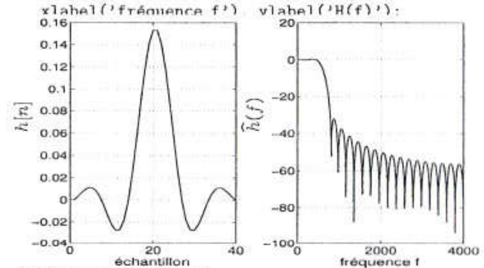
Pour les filtres passe-haut et coupe-bande, leur ordre peut être calculé de la même façon que pour les filtres passe-bas et passe-bande en renversant les fréquences de 0 vers 1 et de 1 vers 0. (e.g. l'ordre d'un passe-haut `Wp=0.2, Ws=0.1` est le même que celui d'un passe-bas `Wp=0.8, Ws=0.0`).

4 Exemple

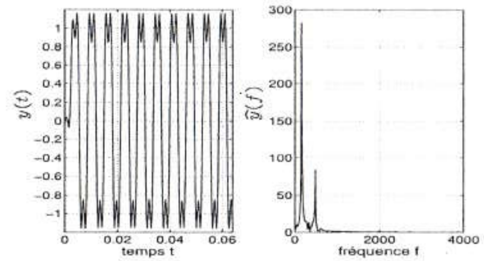
```
% Génération du signal
  Fe = 8e3;
  N = 512;
  t = (0:N-1)/Fe;
  x = square(2*pi*Fe*t/50);
% TFD sur [0, Fe]
  X = fft(x);
  f = (0:N-1)/N*Fe;
% Affichage
  subplot(1,2,1); plot(t,x);
  xlabel('temps t'), ylabel('x(t)');
  subplot(1,2,2); plot(f(1:N/2),abs(X(1:N/2)));
  xlabel('fréquence f'), ylabel('X(f)');
```




```
% Synthèse du filtre passe bas
% (RIF moindres carrés)
% Bande passante [0, 200 Hz]
% Bande coupée [400Hz, 4000Hz]
% Réponse impulsionnelle
  h = firls(39,[0 500 750 Fe/2]/Fe*2,[1 1 0 0]);
% Réponse en frquence
  [H, freq] = freqz(h,1,512,Fe);
% Affichage
  subplot(1,2,1); plot(h);
  xlabel('échantillon'), ylabel('h[n]');
  subplot(1,2,2); plot(freq,20*log10(abs(H)));
  xlabel('fréquence f'), ylabel('H(f)');
```




```
% Filtrage du signal
  y = filter(h,1,x);
  Y = fft(y);
% Affichage
  subplot(1,2,1); plot(t,y);
  xlabel('temps t'), ylabel('y(t)');
  subplot(1,2,2); plot(f(1:N/2),abs(Y(1:N/2)));
  xlabel('fréquence f'), ylabel('Y(f)');
```



— Matlab et le traitement du signal —

Table des matières

Matlab et sa boîte à outils *Signal Processing*, contiennent un grand nombre de fonctionnalités concernant :

- la génération de signaux ;
- la représentation des signaux (Transformée de Fourier Discrète FFT, Transformée en Cosinus Discrets DCT...) ;
- l'analyse des signaux (statistique, analyse spectrale paramétrique...) ;
- la représentation des systèmes linéaires (fonction de transfert, pôles et zéros, espace d'état...) ;
- l'analyse des systèmes (réponse impulsionnelle, réponse en fréquence...) ;
- le filtrage et la synthèse de filtres.

Nous nous intéresserons ici uniquement aux fonctions utiles pour la représentation fréquentielle des signaux et des systèmes linéaires et aux fonctions de filtrage et de synthèse de filtres.

1 Représentation des signaux et systèmes

1.1 Temps

Un signal numérique échantillonné à la fréquence f_e se représente naturellement dans Matlab, comme un vecteur de `N` éléments (signal de durée $\frac{N}{f_e}$). Le vecteur des temps qui lui est associé est :

```
>> t = (0:N-1)/fe;
```

1.2 Autocorrélation

L'estimation de l'autocorrélation d'un signal ou de l'intercorrélation de deux signaux de longueur `N` peut être effectuée avec la fonction `xcorr` :

```
>> Cxy = xcorr(x,y,option);
```

C'est un vecteur de longueur `2N-1` tel que le `N`ième élément corresponde à la corrélation en 0. si `option` n'est pas donné, `xcorr` estime la corrélation non normalisée :

$$C^{\mathrm{un}}_{x,y}[n] = \begin{cases} \sum_{k=1}^{N-n} x^*[k]y[k+n] & \text{si } n \geq 0 \\ C_{x,y}[-n]^* & \text{si } n < 0 \end{cases}$$

`option` peut prendre les valeurs :
- `'biased'` pour l'estimateur biaisé de la corrélation :

$$C^{\mathrm{b}}_{x,y}[n] = \frac{1}{N} C^{\mathrm{un}}_{x,y}[n].$$

- `'unbiased'` pour l'estimateur non biaisé de la corrélation :

$$C^{\mathrm{nb}}_{x,y}[n] = \frac{1}{|N-n|} C^{\mathrm{un}}_{x,y}[n].$$

- `'coeff'` pour laquelle la corrélation est normalisée de façon à ce que $C^{\mathrm{c}}_{x,y}(0) = 1$.

1.3 Fonction de transfert

- La fonction de transfert (transformée de Laplace de la réponse impulsionnelle) d'un filtre analogique s'écrit sous la forme :

$$F(s) = \frac{b_0 s^M + b_1 s^{M-1} + \cdots + b_{M-1}s + b_M}{s^N + a_1 s^{N-1} + \cdots + a_{N-1}s + a_N}$$

ou sous la forme :

$$F(s) == K \frac{\sum_{k=1}^{M}(s - z_k)}{\sum_{k=1}^{N}(s - p_k)}.$$

- La fonction de transfert (transformée en z de la réponse impulsionnelle) d'un filtre numérique s'écrit quant à elle :

$$F(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} = K \frac{\sum_{k=1}^{M}(z - z_k)}{\sum_{k=1}^{N}(z - p_k)}.$$

Ces systèmes peuvent donc se représenter dans Matlab, avec les vecteurs du dénominateur `a=[1,` $a_1, \ldots a_N$`]` et du numérateur `b=[`b_0, $b_1, \ldots b_M$`]` ou par le gain `K` et les vecteurs des poles `p=[`p_0, $p_1, \ldots p_M$`]` et des zéros `z=[`z_0, $z_1, \ldots z_M$`]`.

Un filtre numérique à réponse impulsionnelle finie (RIF) ayant son dénominateur à 1 sera entièrement caractérisé par sa réponse impulsionnelle (`h = b`).

2 Représentations fréquentielles

2.1 Signaux

La Transformée de Fourier Discrète d'un signal de `N` points est calculée par un algorithme rapide (*Fast Fourier Transform* FFT) :

```
>> X = fft(x) ;
```

C'est également un signal (à valeurs complexes) de `N` points échantillonnés à la fréquence $\frac{N}{f_e}$. Le vecteur des fréquences qui lui est associé est :

```
>> f = (0:N-1)/N*fe;
```

Rappelons que ce signal est de période f_e ; on peut le représenter sur l'intervalle $[-\frac{f_e}{2}, \frac{f_e}{2}]$ grâce à la fonction `fftshift` (qui ne fait qu'un décalage des vecteurs et aucun calcul de fft) :

```
>> Y = fftshift(X);
```

Le vecteur des fréquences qui lui est associé est alors :

```
>> f = (0:N-1)/N*fe - fe/2;
```

2.2 Systèmes

La réponse en fréquence d'un système analogique est donnée par :

```
>> H=freqs(b,a,w);
```

`H` est la réponse en fréquence aux pulsations données dans le vecteur `w` (en radian par seconde).

La réponse en fréquence d'un système numérique est donnée par :

```
>> H=freqz(b,a,f,fe);
```

`H` est la réponse du système aux fréquences données dans le vecteur `f` (en Hertz) et `fe` la fréquence d'échantillonnage.

Voir l'aide en ligne pour plus de détails...

Remarque : Matlab, travaille en pulsation pour les systèmes analogiques et en fréquence (et même en fréquence normalisée) pour les systèmes numériques.

3 Filtrage et synthèse de filtres

3.1 Filtrage

Le filtrage du vecteur `x` par le filtre numérique défini par `a` et `b` est effectué par :

```
>> y = filter(b,a,x);
```

Remarque : Les conditions initiales de l'équation de récurence peuvent être données en entrée de la fonction `filter`. Elles se calculent par la fonction `filtic`.

3.2 Synthèse filtres RIF

Il existe différentes méthodes de synthèse de filtres RIF approchant un filtre idéal :

3.2.1 Troncature de la Réponse Impulsionnelle

La fonction `fir1` synthétise un filtre RIF simple (défini par une seule bande passante ou coupée) par troncature et fenêtrage de la réponse impulsionnelle du filtre numérique idéal :

```
>> h = fir1(n,fn,type,window) ;
```

- `n` est l'ordre du filtre (longueur de la RI moins un).
- Les fréquences `fn` sont normalisées par rapport à la fréquence de Nyquist (`fn`= $f/\frac{f_e}{2}, 0 \leq$ `fn` ≤ 1). `fn` indique la fréquence de coupure pour les passe-bas et passe-haut, et les fréquences de coupures basse et haute pour les passe-bande et coupe-bande.
- La chaîne de caractère `type` précise le type de filtre. `'high'` pour passe-haut, `'stop'` pour coupe-bande, type omis pour les passe-bas et passe-bande.
- Le vecteur `window` de longueur `n+1`, correspond à la fenêtre prise en compte (par défaut fenêtre de Hamming). Les fonctions Matlab, disponibles pour créer des fenêtres sont : `bartlett`, `blackman`, `boxcar` (rectangulaire), `chebwin` (chebychev), `Hamming`, `hanning`, `kaiser`, `triang` (triangulaire).

Voir l'aide en ligne pour plus de détails...

3.2.2 Échantillonnage de la Réponse en Fréquence

La fonction `fir2` synthétise un filtre RIF par échantillonnage de la réponse en fréquence du filtre analogique idéal et fenêtrage de la réponse impulsionnelle du filtre ainsi construit.

```
>> h = fir2(n,fn,m,window) ;
```

- `n` est l'ordre du filtre (longueur de la RI moins un).
- `fn` est le vecteur des fréquences normalisées ($0 \leq$ `fn` ≤ 1) définissant le filtre idéal comme linéaire par morceaux.
- `m` est le vecteur des amplitudes, aux fréquences données par `fn`, de la réponse en fréquence du filtre idéal.

Voir l'aide en ligne pour plus de détails...

3.2.3 Moindres Carrés

La fonction `firls` synthétise un filtre RIF approchant au mieux, au sens des moindres carrés (norme L_2), la réponse en fréquence du filtre analogique idéal.

```
>> h = firls(n,fn,m) ;
```

- `n` est l'ordre du filtre (longueur de la RI moins un).
- `fn` est le vecteur des fréquences normalisées ($0 \leq$ `fn` ≤ 1) définissant le filtre idéal.
  Attention, contrairement à `fir2`, ces fréquences sont prises deux par deux dans `firls`, permettant ainsi de définir des bandes de fréquences ou le filtre idéal n'est pas précisé (bandes de transition).

- `m` est le vecteur des amplitudes de la réponse en fréquence du filtre idéal aux fréquences `fn`.

Voir l'aide en ligne pour plus de détails...

3.2.4 Méthode de Rémez

La fonction `remez` synthétise un filtre RIF approchant au mieux, au sens du minimax (norme L_∞), la réponse en fréquence du filtre idéal.

```
>> h = remez(n,fn,m) ;
```

Les paramètres sont les mêmes que pour `firls`.
La fonction `remezord` permet de plus d'estimer l'ordre nécessaire à la méthode de remez pour construire un filtre de déviation maximale donnée.
Voir l'aide en ligne pour plus de détails...

3.3 Synthèse de filtres RII

Les principales méthodes de synthèse de filtres à réponse impulsionnelle infinie (RII) procèdent par discrétisation d'un filtre analogique.

3.3.1 Synthèse de filtres analogiques passe-bas

Les fonctions suivantes renvoient les pôles (`p`) zéros (`z`) et gain (`k`) des filtres analogiques passe-bas normalisés (pulsation de coupure unité) :

```
% Butterworth
>> [z, p, k] = buttap(n) ;
% Chebychev
% Oscillations inférieures à Rp dB
% en bande passante
>> [z, p, k] = cheb1ap(n, Rp) ;
% Oscillations au deçà de Rs dB
% en bande coupée
>> [z, p, k] = cheb2ap(n, Rs) ;
% Elliptique : oscillations inférieures
% à Rp dB en bande passante et au deçà
% de Rs dB en bande coupée
>> [z, p, k] = ellipap (n, Rp, Rs) ;
```

Pour obtenir une représentation de ces filtres analogiques en terme des numérateurs et dénominateurs de leur fonction de transfert (transformée de Laplace de leur réponse impulsionnelle) :

```
>> [b, a] = zp2tf(z,p,k) ;
```

3.3.2 Transformation des fréquences

Pour transformer les filtres passe-bas en tout type de filtres :

`% Lowpass -> Lowpass` $(p \longrightarrow \frac{p}{\omega_0})$
`>> [bt, at] = lp2lp(b,a,W0) ;`

`% Lowpass -> Highpass` $(p \longrightarrow \frac{\omega_0}{p})$
`>> [bt, at] = lp2hp(b,a,W0) ;`

`% Lowpass -> Bandpass` $(p \longrightarrow \frac{1}{B}(\frac{p}{\omega_0} + \frac{\omega_0}{p}))$
`>> [bt, at] = lp2bp(b,a,W0,Bw) ;`

`% Lowpass -> bandstop` $(p \longrightarrow B\frac{1}{\frac{p}{\omega_0}+\frac{\omega_0}{p}})$
`>> [bt, at] = lp2bs(b,a,W0,Bw) ;`

Si ω_b est la pulsation basse de coupure et ω_h la pulsation haute de coupure, alors la pulsation propre du filtre ω_0 et la largeur de bande du filtre sont donnés par : $B = \omega_h - \omega_b$ et $\omega_0 = \sqrt{\omega_h \omega_b}$.

3.3.3 Discrétisation des filtres

La discrétisation des filtres analogiques permet d'obtenir les coefficients des filtres numériques à partir de ceux du filtre analogiques. Deux techniques sont disponibles à cette fin dans Matlab :

- Discrétisation par invariance de la réponse impulsionnelle :
  ```
  >> [bd, ad] = impinvar(b, a, fe) ;
  ```
  Où `fe` est la fréquence d'échantillonnage.
- Discrétisation par transformation bilinéaire : (approximation $p \approx \frac{2}{T_e}\frac{1-z^{-1}}{1+z^{-1}}$)
  ```
  >> [bd, ad] = bilinear(b, a, fe) ;
  ```
  Attention, la transformation bilinéaire provoque une déformation des fréquences (soit f_a la fréquence analogique et f_n la fréquence numérique) :
  $$f_n = \frac{f_e}{\pi}\arctan(\frac{\pi f_a}{f_e})$$
  et
  $$f_a = \frac{f_e}{\pi}\tan(\frac{\pi f_n}{f_e}).$$
  Il est donc nécessaire de pré-déformer le gabarit du filtre analogique pour obtenir le filtre numérique désiré.

3.3.4 Synthèse complète des filtres

Matlab, propose des fonctions dans lesquelles la synthèse complète du filtre numérique est effectuée :

```
% Butterworth
>> [b, a] = butter(n,Wn,type) ;
% Chebychev
% Oscillations de Rp dB en bande
% passante
>> [b, a] = cheby1(n,Rp,Wn,type) ;
% Oscillations au deçà de Rs dB
% en bande coupée
>> [b, a] = cheby2(n,Rs,Wn,type) ;
% Elliptique : oscillations de Rp dB en
% bande passante et au deçà de Rs dB en
% bande coupée
>> [b, a] = ellip(n,Rp,Rs,Wn,type) ;
```

Ces fonctions donnent directement les coefficients `a` et `b` du filtre numérique à partir de :
- `n` ordre du filtre.