Chapitre n°1
Traitement Numérique du Signal

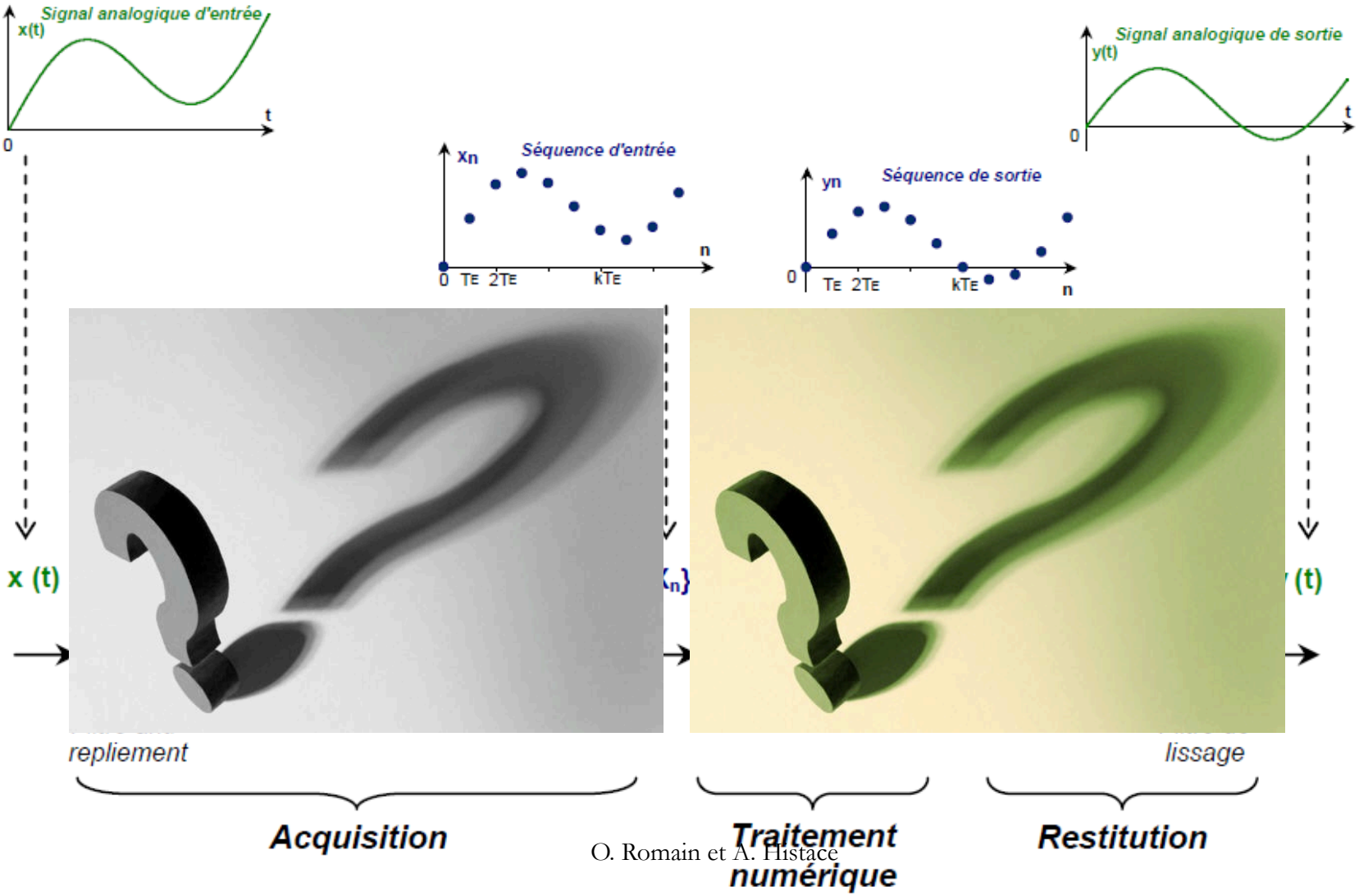
Aymeric Histace
Professeur des Universités
aymeric.histace@ensea.fr
aymeric.histace.free.fr

Plan du cours

Partie 1

Classification des signaux

Chaine de traitement numérique du signal



Introduction

- Un signal est la représentation physique de l'information qu'il transporte. C'est un vecteur d'information
- Un signal est généralement la transposition électrique (courant ou tension) d'une grandeur mesurable (pression, onde acoustique, etc.)
- Le traitement du signal a pour principaux objectifs l'interprétation, l'extraction et l'élaboration des signaux

Introduction

- Le traitement du signal peut être analogique (continue) ou numérique (discret)
  - TS : Traitement du signal
    - signaux analogiques
  - TNS : Traitement Numérique du Signal
    - le signal analogique est converti en numérique
- Le traitement du signal nécessite de connaître a priori le signal à traiter pour dimensionner la chaîne de traitement

Propriété des signaux

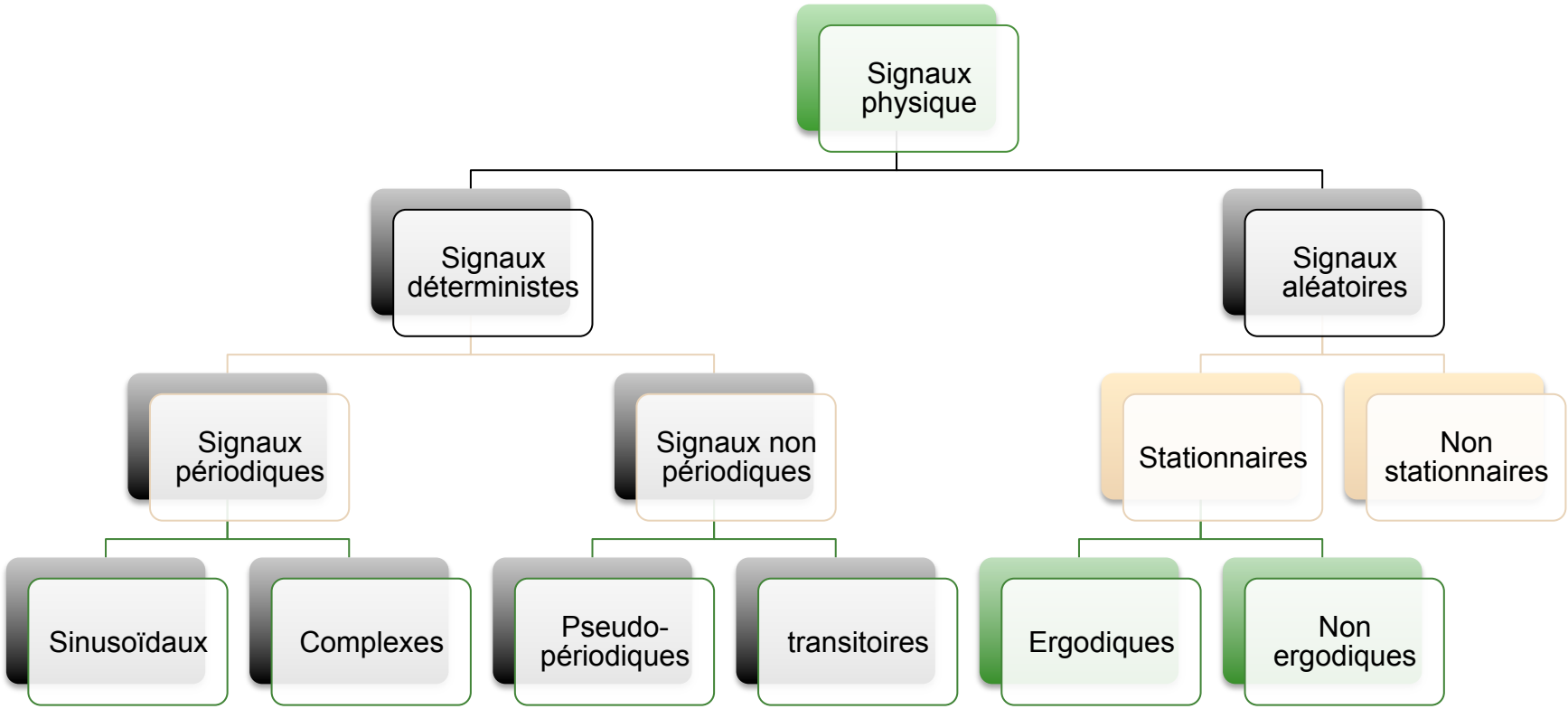
- Les signaux sont représentés par des fonctions à valeurs réelles d'une variable réelle t.

- Un signal peut posséder les propriétés suivantes :
  - Energie bornée ou infinie
  - Stationnaire ou pas
  - Amplitude bornée ou non
  - Continu temporellement ou avec discontinuités
  - Causal (e(t)=0 pour t<0) ou non causal
  - Spectre bornée ou infinie
  - Valeurs réelles ou complexes
  - Déterministe ou probabiliste

Classification de signaux

- Il existe plusieurs façons de classer les signaux
  - Caractéristiques temporelles
  - Représentations fréquentielles
  - Caractéristiques énergétiques
  - Caractéristiques morphologiques (continues ou discrètes)

Classification de signaux / temporel



Classification spectrale

- Un signal peut être classé en fonction de la répartition de sa puissance en fréquence. Le domaine occupé par son spectre B est aussi appelé la largeur de bande du signal.

- Cette caractéristique en Hz est absolue
- Fmoy=(Fmax-Fmin)/2;

- Signaux à bandes étroites : B/Fmoy < 1
- Signaux à larges bandes : B/Fmoy >> 1

Classification énergétique

- Energie d'un signal : $E(t_1,t_2)=\int_{t_1}^{t_2} s^2(t)\,dt$
- Puissance d'un signal : $P(t_1,t_2)=\frac{1}{t_2-t_1}\int_{t_1}^{t_2} s^2(t)\,dt$
- 2 catégories
  - Signaux à énergie finie (signaux physiques)
  - Signaux à puissance moyenne finie (signaux périodiques ou quasi-périodiques)

Classifications des signaux

Temps

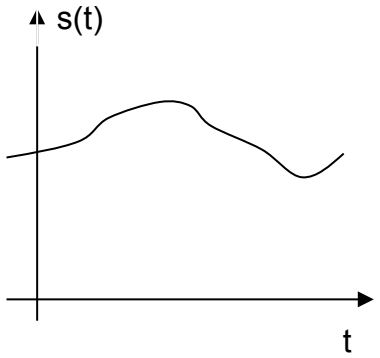
continu

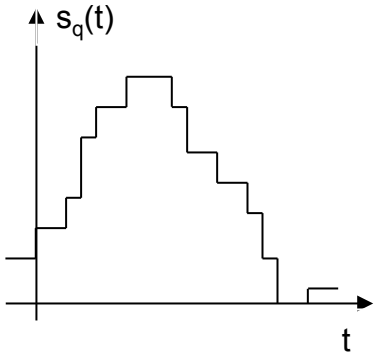
discret

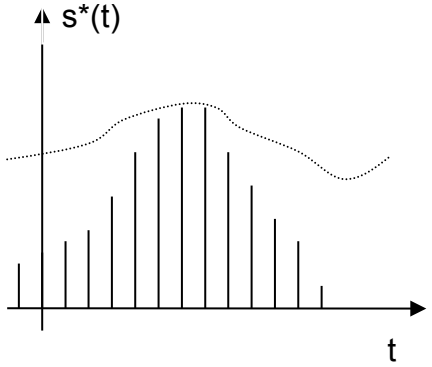
Amplitude

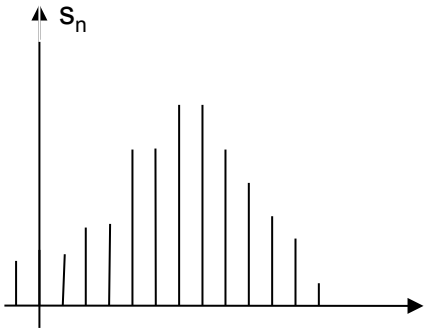
continu

discret









Signaux numériques

- Signal numérique N(t=t1)=(01110010), N(t=t2)=(01110111), …, N(t=t3)=(01111000) est un signal doublement discrétisé
  - Discrétisation en amplitude : quantification
    - Représentation d'une valeur en binaire
  - Discrétisation en temps : échantillonnage

- **Définition :** On appelle numérisation d'un signal les opérations qui consistent à transformer un signal continu en un signal discrétisé temporellement et quantifié en amplitude

Rappels de mathématiques
Transformée de Fourier

- Transformation de Fourier des fonctions périodiques : série de Fourier

- Définition : Théorème de Fourier
  - Si s(t) est une **fonction périodique**, de période To, elle peut se décomposée sous la forme d'une somme de fonctions sinusoïdales de fréquences f multiple de la fréquence Fo, dite fondamentale.

$$s(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n . \cos(2.\pi.n.F_0.t) + b_n . \sin(2.\pi.n.F_0.t) \right]$$

  - avec

$$a_o = \frac{1}{T_0} . \int_0^{T_0} s(t).dt \qquad a_n = \frac{2}{T_0} . \int_0^{T_0} s(t).\cos(2.\pi.F_0.t)dt \qquad b_n = \frac{2}{T_0} . \int_0^{T_0} s(t).\sin(2.\pi.F_0.t)dt$$

Série de Fourier

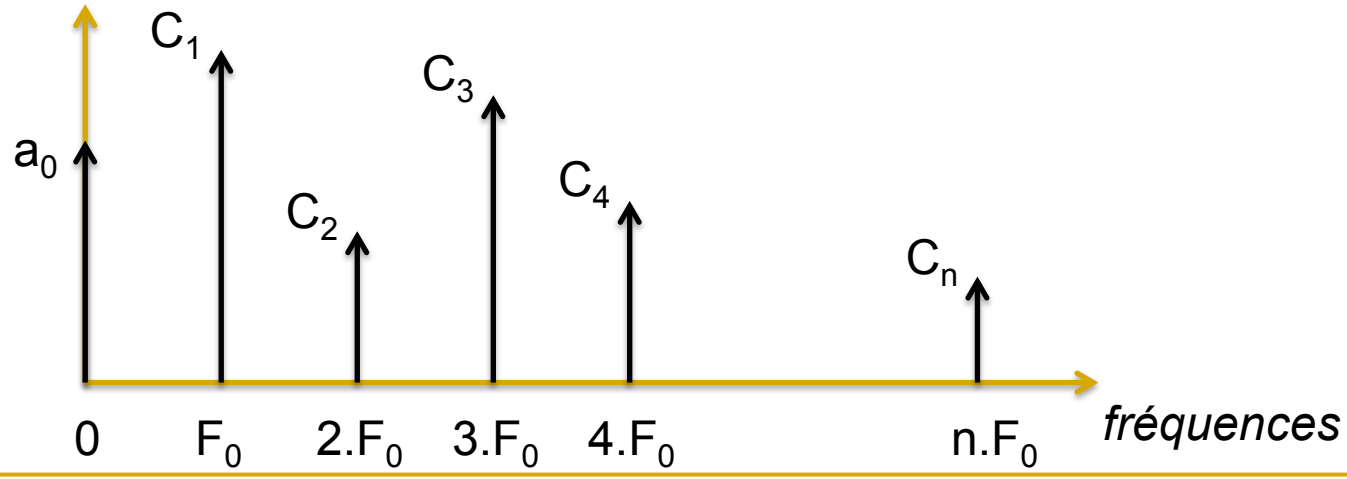
- Autre représentation :

$$s(t) = a_0 + \sum_{n=1}^{\infty} \left[C_n . \cos(2.\pi.n.F_0.t + \varphi_n) \right]$$

$$C_n = \sqrt{a_n^2 + b_n^2}$$

$$\varphi_n = \arctan(-\frac{b_n}{a_n})$$

- Représentation spectrale

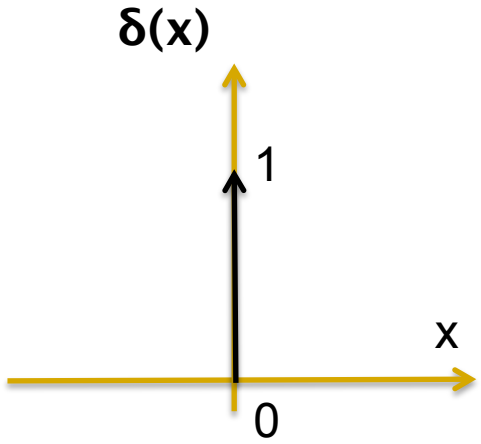


Remarques

- Signaux périodiques
  - Bande occupée : infinie
  - Spectre discret de raies aux fréquences multiple du fondamental
  - Signaux à énergie infinie mais puissance moyenne finie
  - Identité de Parseval

Impulsion de Dirac

- L'impulsion de Dirac est un signal au sens des distributions



$$\int_{-\infty}^{+\infty} \delta(x)\,dx = 1$$

Transformée de Fourier
Cas des signaux non périodiques

- Définition de TF :
  - La Transformée de Fourier TF d'un signal non périodiques est définie par :

$$x(t) \xrightarrow{TF} X(f) \qquad X(f) = \int_{-\infty}^{+\infty} x(t).e^{-j.2.\pi.f.t}\, dt$$

- Définition de TF^{-1} :
  - La Transformée de Fourier inverse TF^{-1} d'un spectre est définie par :

$$x(t) = \int_{-\infty}^{+\infty} X(f).e^{j.2.\pi.f.t}\, df \qquad x(t) \xleftarrow{TF^{-1}} X(f)$$

- Conditions d'existence de TF:
  - Un signal x(t) admet une TF si :
    - x(t) est une fonction bornée
    - L'intégrale de x(t) entre $-\infty$ et $+\infty$ ait une valeur finie
    - x(t) soit une fonction de carré sommable

Propriétés de la TF

- Linéarité

$$x(t) = a.y(t) + b.z(t) \xrightarrow{TF} X(f) = a.Y(f) + b.Z(f)$$

- Homothétie

$$x(a.t) \xleftrightarrow{TF} \frac{1}{|a|} X(\frac{f}{a})$$

- Translation ou retard

$$x(t-a) \xleftrightarrow{TF} X(f).e^{-j2.\pi.a.f}$$

$$x(t)..e^{j2.\pi.f_0.t} \xleftarrow{TF^{-1}} X(f-f_0)$$

Propriétés de la TF

- Dérivation 1er ordre

$$\frac{dx(t)}{dt} \xrightarrow{TF} (j.2.\pi.f).X(f)$$

- Dérivation ordre n

$$\frac{d^n x(t)}{dt^n} \xrightarrow{TF} (j.2.\pi.f)^n.X(f)$$

- Multiplication

$$y(t) = x(t).h(t) \quad \underset{TF^{-1}}{\overset{TF}{\rightleftarrows}} \quad Y(f) = X(f) * H(f)$$

- Convolution

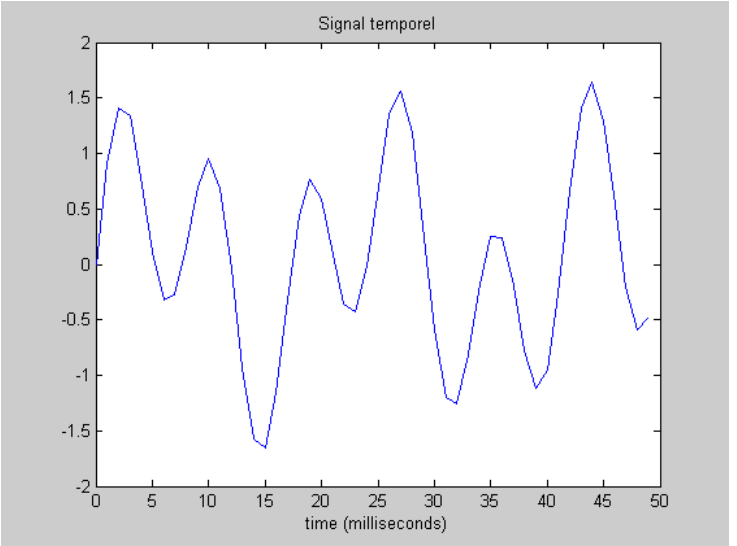
$$y(t) = x(t) * h(t) \quad \underset{TF^{-1}}{\overset{TF}{\rightleftarrows}} \quad Y(f) = X(f).H(f)$$

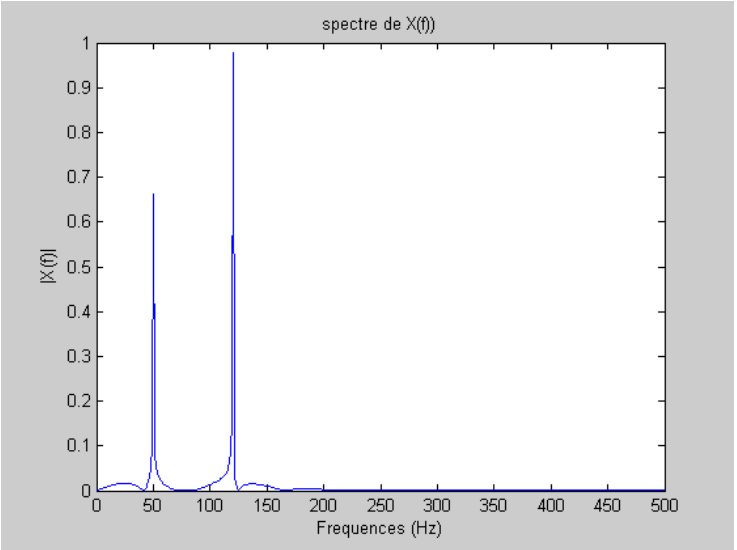
Matlab

- Outils de simulation Mathématique
  - Comme Mapple ou Mathematica
  - Spécialisé dans le calcul matriciel
  - Basé sur des bibliothèques spécialisées (TS, finance, RF, etc.)
- Différents outils inclus
  - Simulink
  - Passerelle vers des cibles de traitement (DSP, FPGA, etc.)
  - Passerelle vers du langage C
- Option Systèmes Embarqués
  - Outil de conception et de test rapide des filtres numériques

Matlab
Exemple de TF

```
clear all;
close all;
clc;
Fs = 1000;
T = 1/Fs;
L = 1000;
t = (0:L-1)*T;
% Somme de deux sinusoïdes de fréquence 50 et 120 Hz
x = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
% Analyse temporelle
plot(Fs*t(1:50),x(1:50))
title('Signal temporel')
xlabel('time (milliseconds)')

%Analyse spectrale
NFFT = 2^nextpow2(L); % Next power of 2 from length of y
Y = fft(x,NFFT)/L;
f = Fs/2*linspace(0,1,NFFT/2+1);

figure
plot(f,2*abs(Y(1:NFFT/2+1)))
title('spectre de X(f))')
xlabel('Frequences (Hz)')
ylabel('|X(f)|')
```

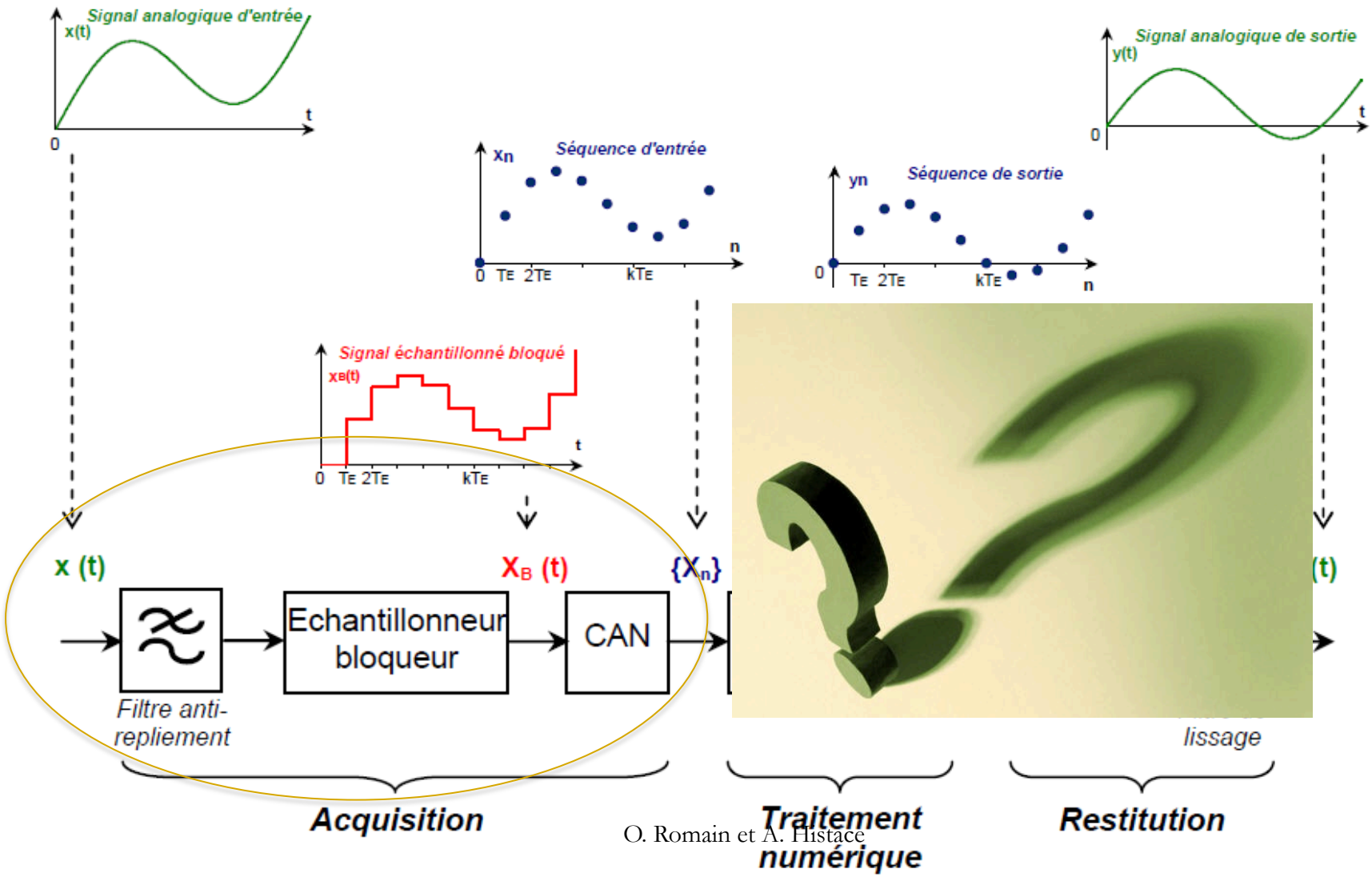




Partie 2

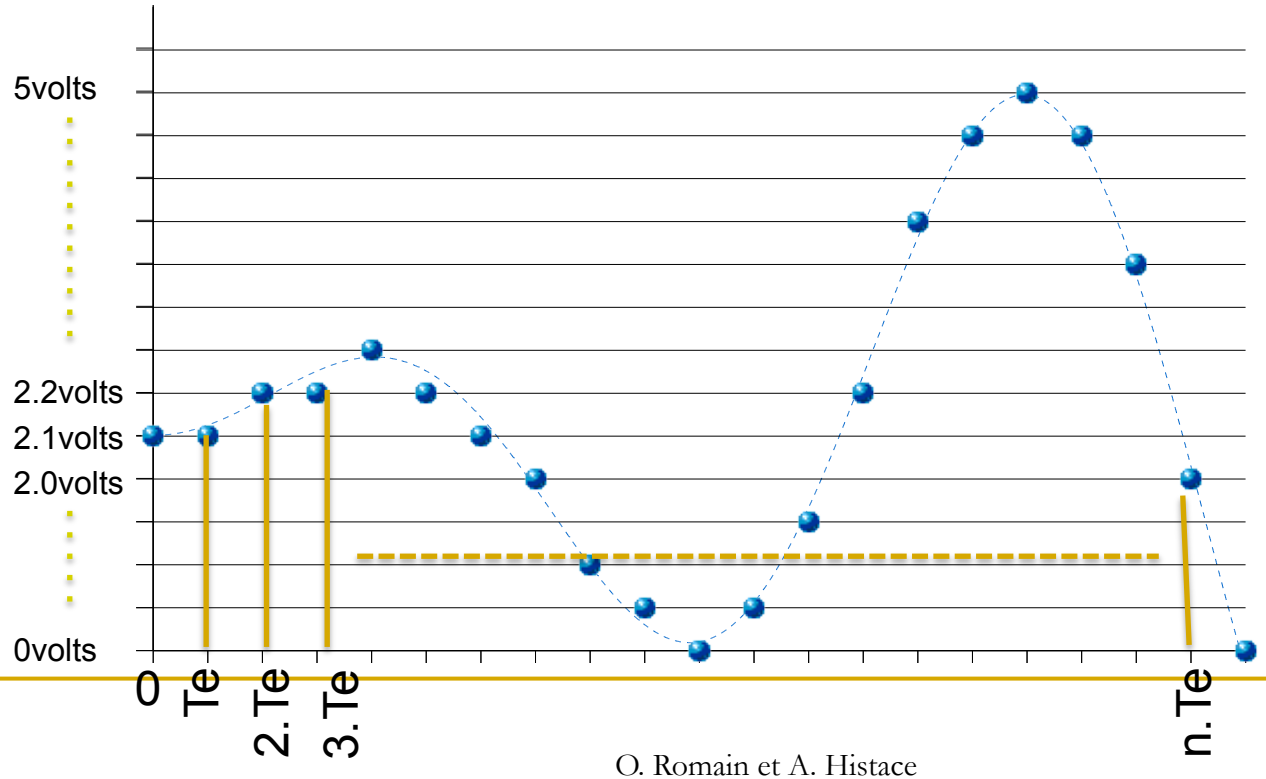
Echantillonnage

Chaine de traitement numérique du signal



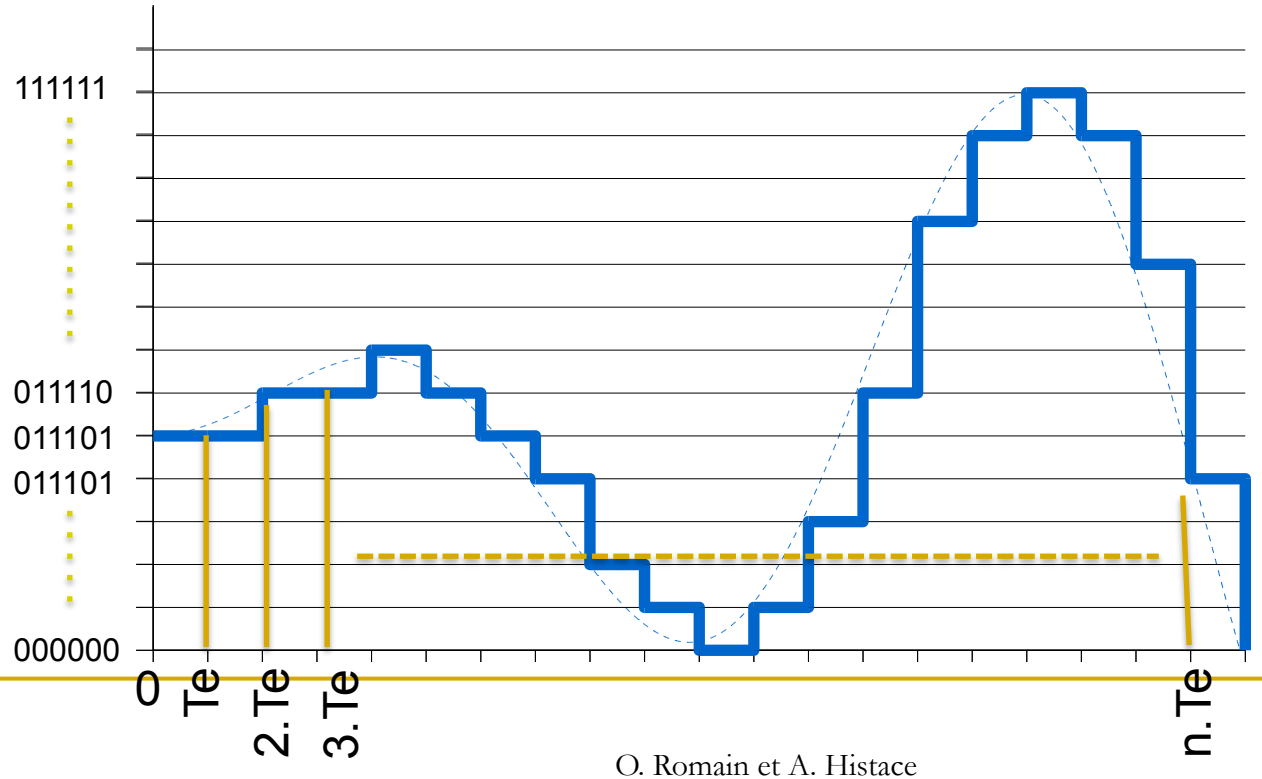
Echantillonnage / sampling

- L'échantillonnage consiste à représenter un signal analogique continu s(t) par un ensemble de valeurs discrètes s(nTe) avec n entier et **Te** constant appelé **période d'échantillonnage**

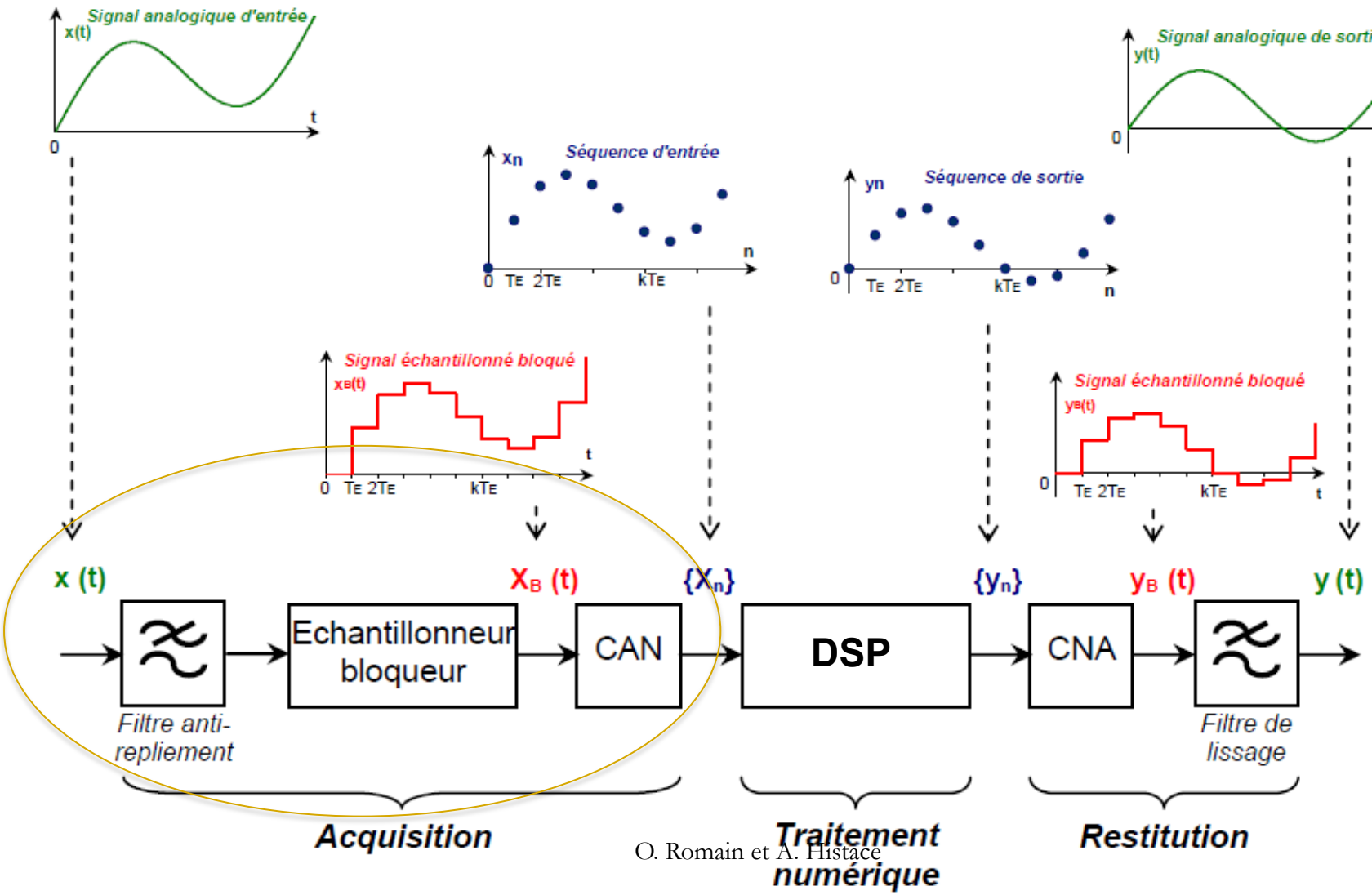


Echantillonnage / numérisation

- La numérisation d'un signal consiste en l'échantillonnage et la quantification des amplitudes du signal échantillonné. Le signal numérisé est discrétisé en temps et en amplitude. Il correspond à un signal numérique (binaire) qui évolue au rythme de Te.

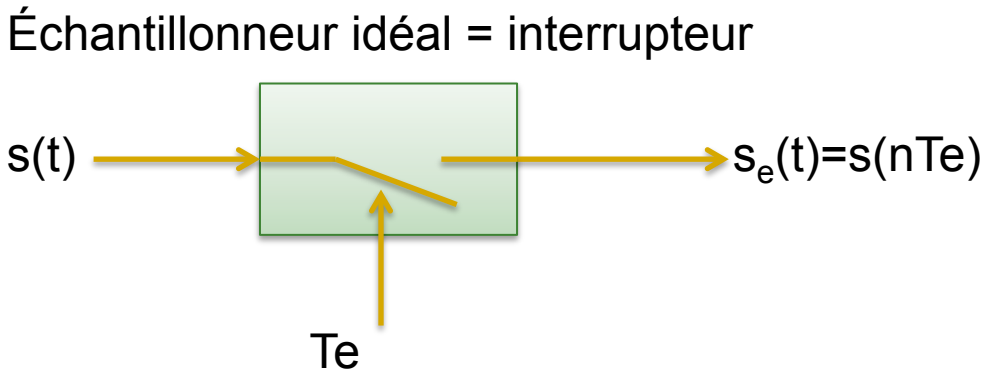


Chaine de traitement numérique du signal

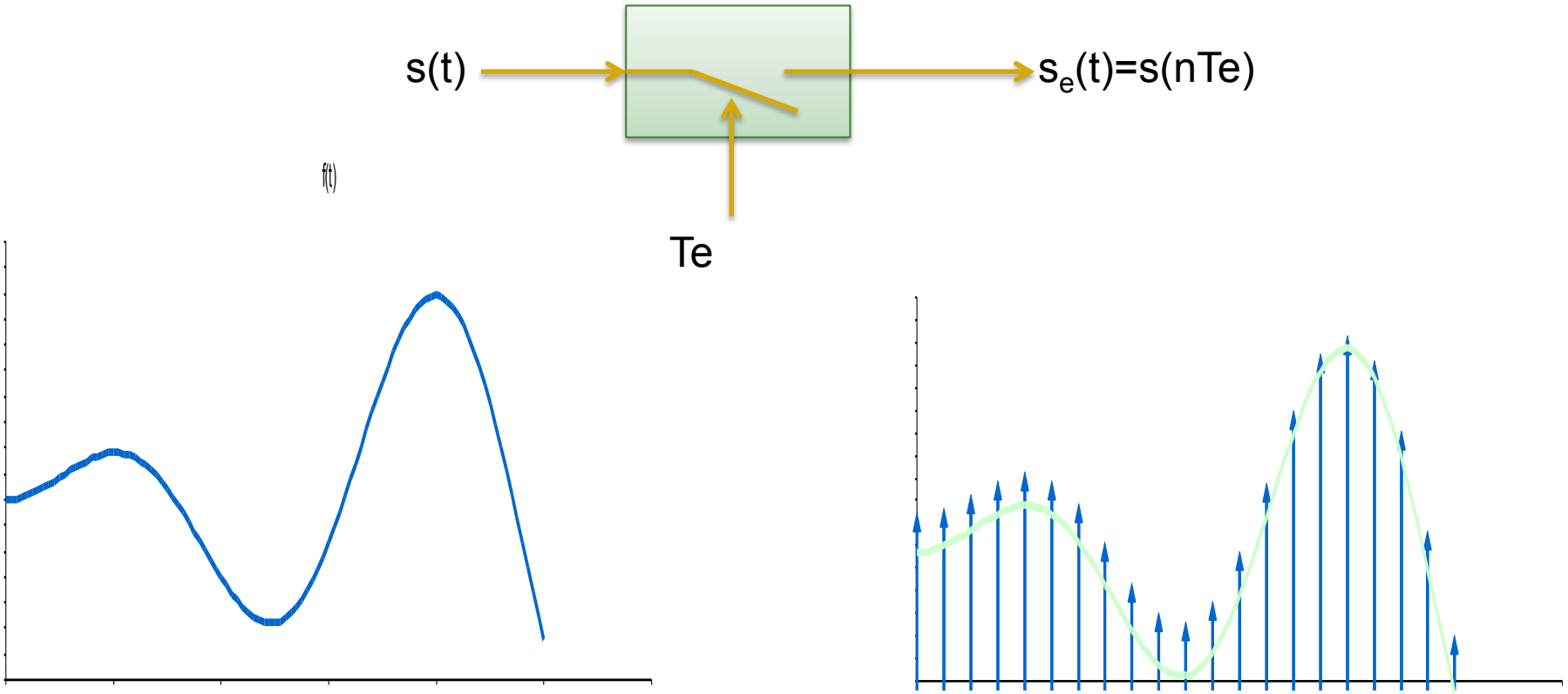


Echantillonnage idéal

- Définition :
  - On suppose que le signal s(t) a un spectre à support borné, c'est à dire que le spectre de S(f) est limité (S(f) =0 pour f>fmax). Cette limitation est soit naturelle, soit artificielle par l'utilisation d'un filtre qui réduit l'encombrement spectrale.
  - L'échantillonnage correspond à l'opération de prélevé à période constante l'amplitude du signal d'entrée.



Echantillonnage idéal



Echantillonnage idéal

- Définition :
  - Le signal échantillonnée correspond à multiplier le signal d'entrée par un peigne de dirac

$$s_e(t) = s(t) \cdot \sum_{k=-\infty}^{+\infty} \delta(t - kT_e)$$

- Définition :
  - Un peigne de dirac est : $Pgn_{Te}(t) = \sum_{k=-\infty}^{+\infty} \delta(t - kT_e)$
  - La transformée de Fourier du peigne de dirac est :

$$Pgn_{Te}(t) \xrightarrow{TF} \frac{1}{T_e} \cdot \sum_{k=-\infty}^{+\infty} \delta(f - \frac{k}{T_e})$$

Echantillonnage idéal / analyse spectrale

- Quel est le spectre du signal échantillonné ?

$$s_e(t) = s(t) \cdot \sum_{k=-\infty}^{+\infty} \delta(t - kT_e) = s(t).Pgne_{T_e}(t)$$

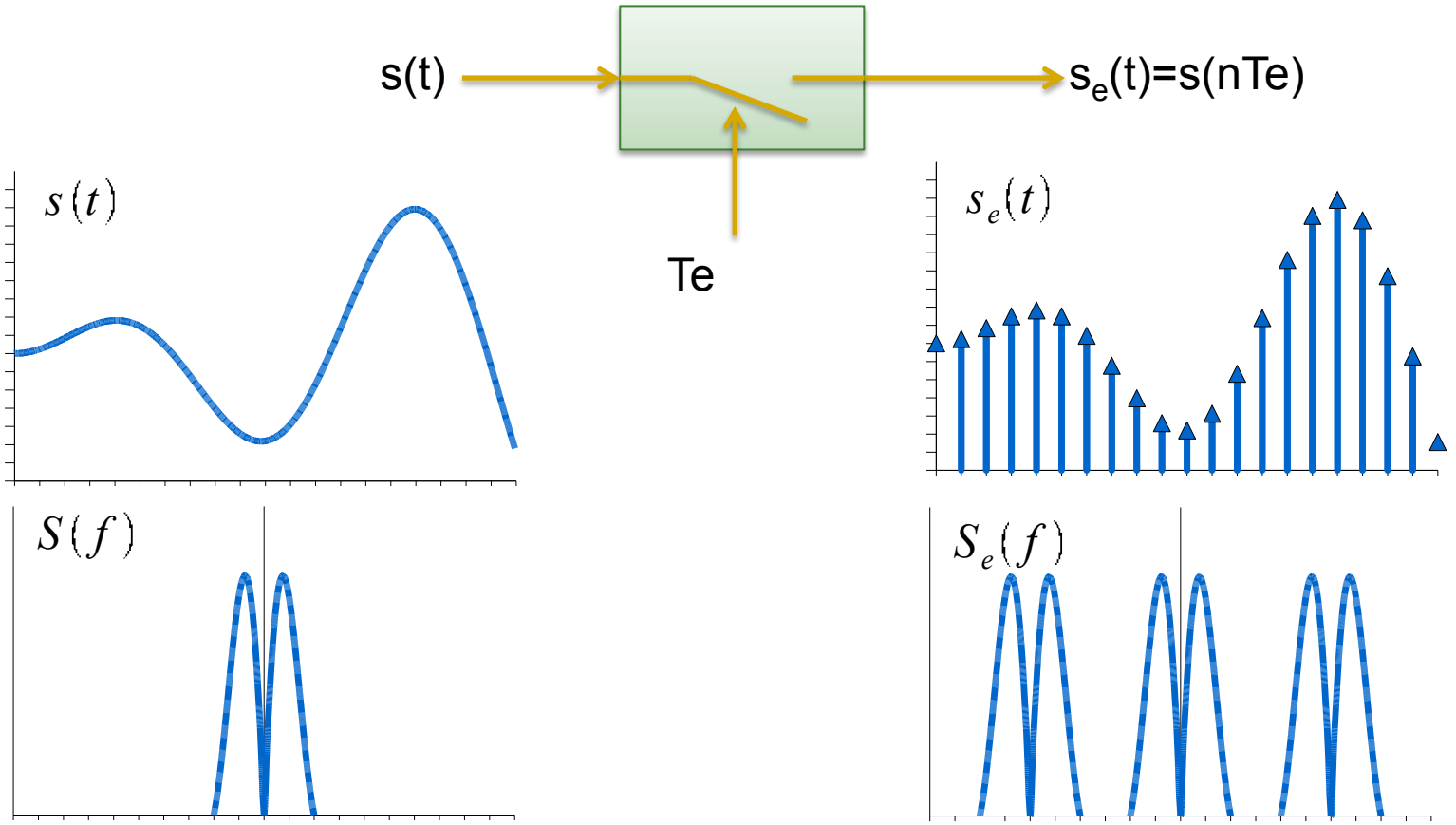
TF

$$S_e(f) = TF(s(t)) * TF(Pgn_{Te}(t)) = S(f) * \frac{1}{T_e} \sum_{k=-\infty}^{+\infty} \delta(f - k.f_e)$$

$$S_e(f) = S(f) * \frac{1}{T_e} \sum_{k=-\infty}^{+\infty} \delta(f - k.f_e) = \frac{1}{T_e} \sum_{k=-\infty}^{+\infty} S(f - k.f_e)$$

- Définition
  - Le spectre d'un signal échantillonné correspond à la périodisation du spectre du signal d'entrée

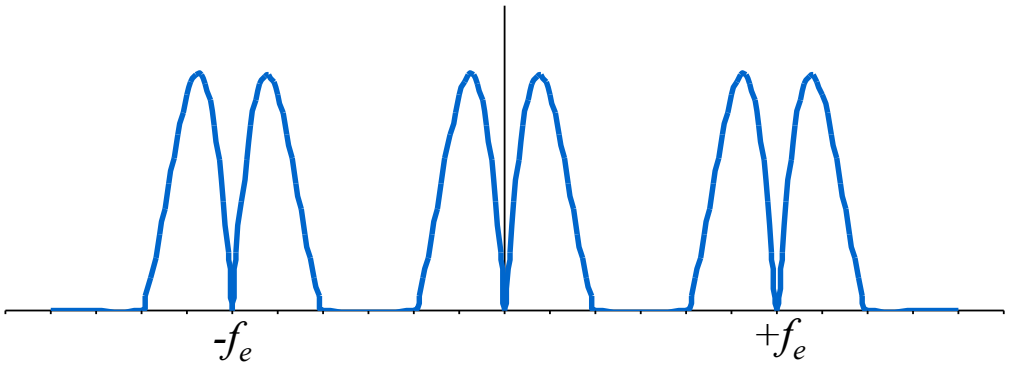
Echantillonnage idéal



Périodisation du spectre

Echantillonnage idéal

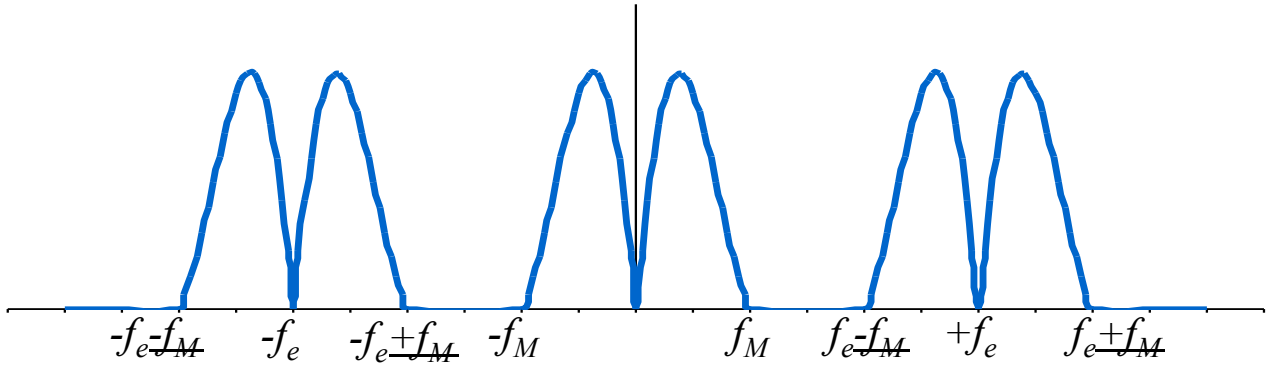
- Remarque :
  - Le spectre du signal d'entrée est borné
  - Le spectre d'un signal échantillonné est infinie



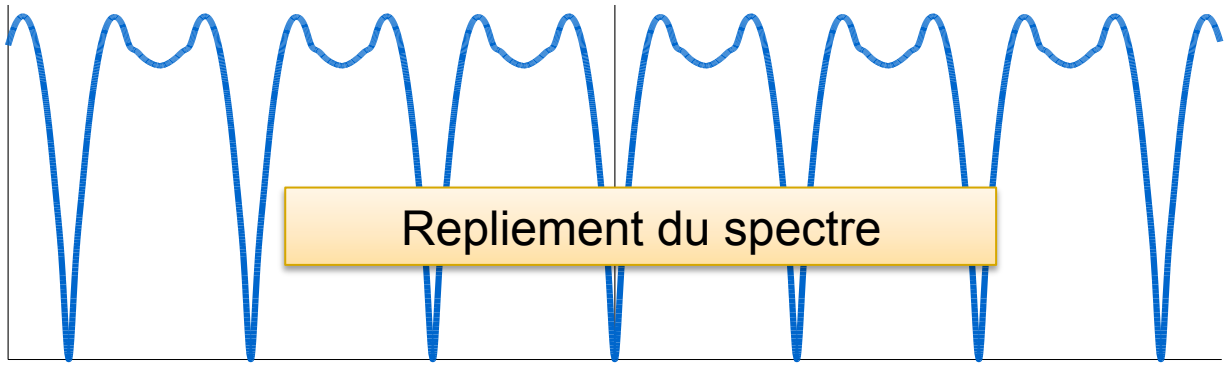
$$S_e(f) = f_e \cdot \sum_{n=-\infty}^{+\infty} S(f - f_e)$$

Echantillonnage idéal/ Fe ?

- Comment choisir la fréquence d'échantillonnage Fe ?



$f_e > 2.f_M$



$f_e < 2.f_M$

Echantillonnage / Fe ?

- Fréquence d'échantillonnage trop basse, le spectre du signal échantillonné est replié sur lui même

- Théorème de Shanon ou Nyquist:
  - La fréquence d'échantillonnage doit être 2 fois supérieure à la bande B occupée par le signal original

$$f_e \geq 2.B$$

  - Fe=2.B représente la fréquence critique d'échantillonnage ou Nyquist
  - Si Fe>B ⇔sur-échantillonnage
  - Si Fe<B ⇔sous-échantillonnage

- Si les conditions sur la fréquence ne sont pas respectées, le spectre du signal échantillonné sera replié.

Exemple : choix de Fe

$s(t)=\sin(2\pi f_0 t)$ Considérons 3 choix de période d'échantillonnage Te

1er choix : $T = \frac{1}{f_0}$

$T_e = \frac{T}{8}$ $f_e = 8\ f_0$

2ème choix :

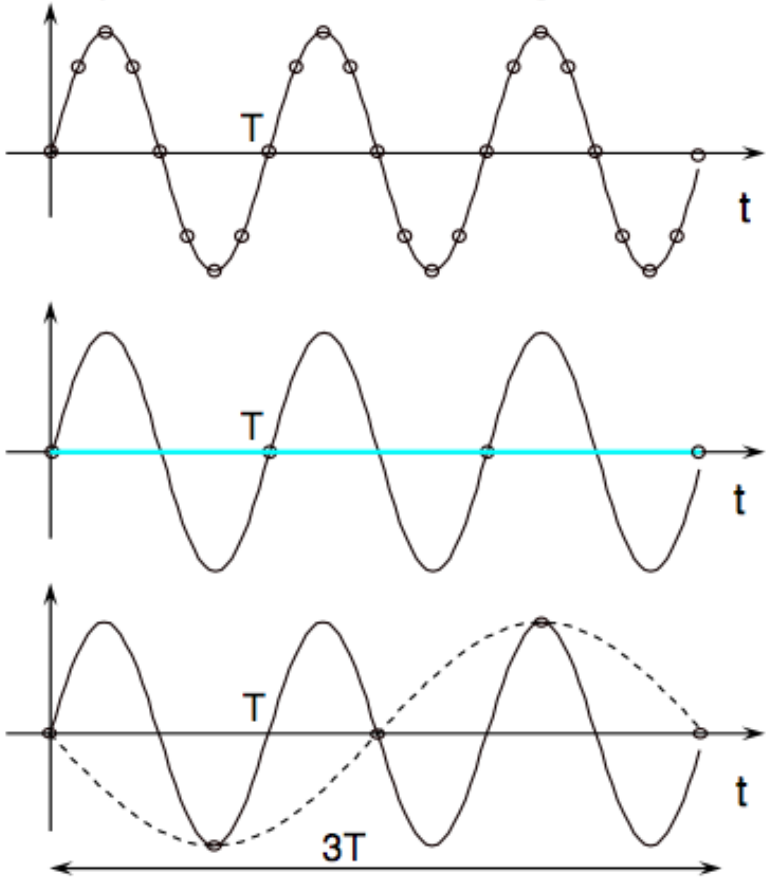
$T_e = T$ $f_e = f_0$

3ème choix :

$T_e = \frac{3}{4} T$ $f_e = \frac{4}{3}\ f_0$

→ ***Repliement spectral***

$f = f_e - f_0 = \frac{1}{3}\ f_0$



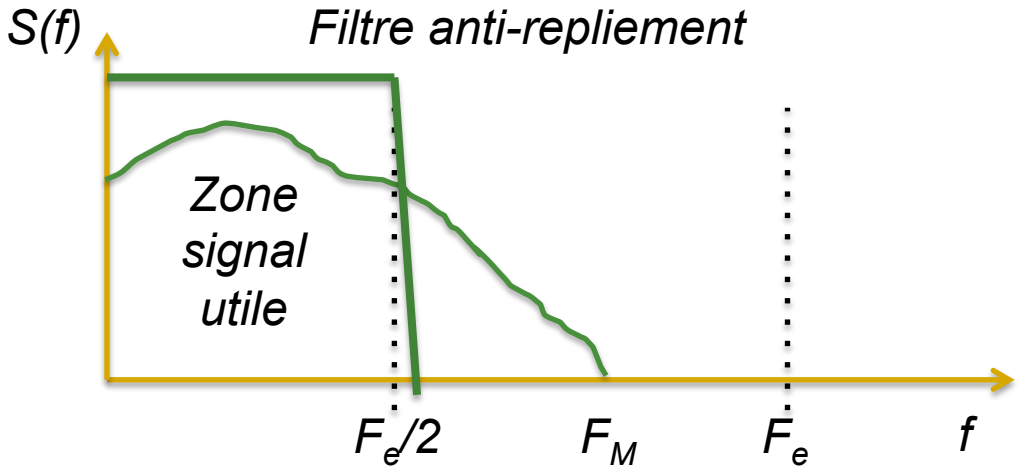
Filtre anti-repliement

- Pour éviter les répliques indésirables dues au repliement, il est indispensable que le spectre du signal d'entrée ne dépasse pas la fréquence de Nyquist : Fe/2

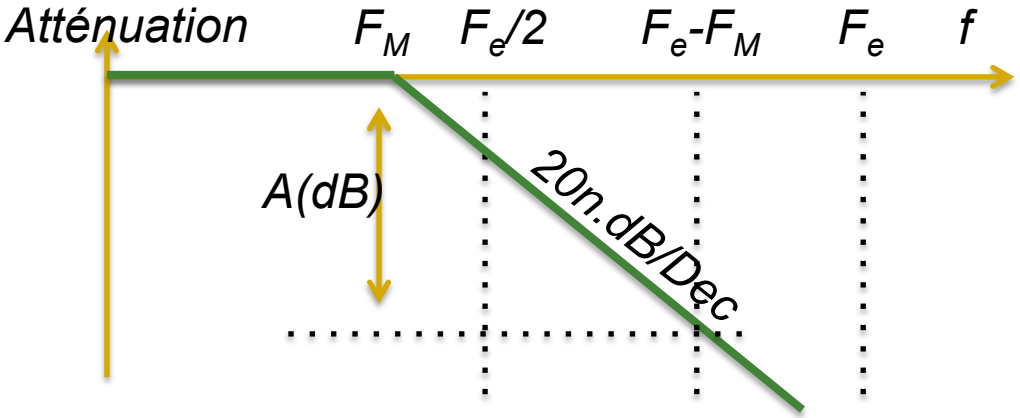
- Si le signal d'entrée a un spectre dont les fréquences sont supérieures à la fréquence de Nyquist, un filtre analogique avant le CAN doit être utilisé pour limiter la bande de s(t)

Filtre anti-repliement

- Filtre théorique :
  - Pente infinie à Fe/2

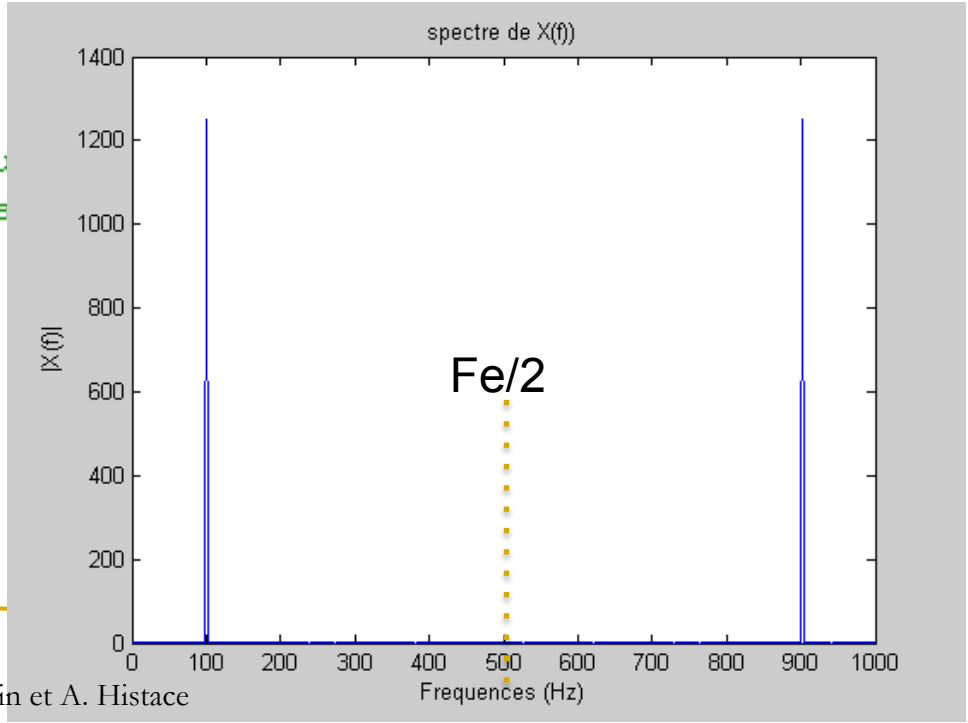


- Filtre pratique :
  - Atténuation de A dB à F_e-F_M
  - Pente du filtre nécessaire



Matlab : exemple

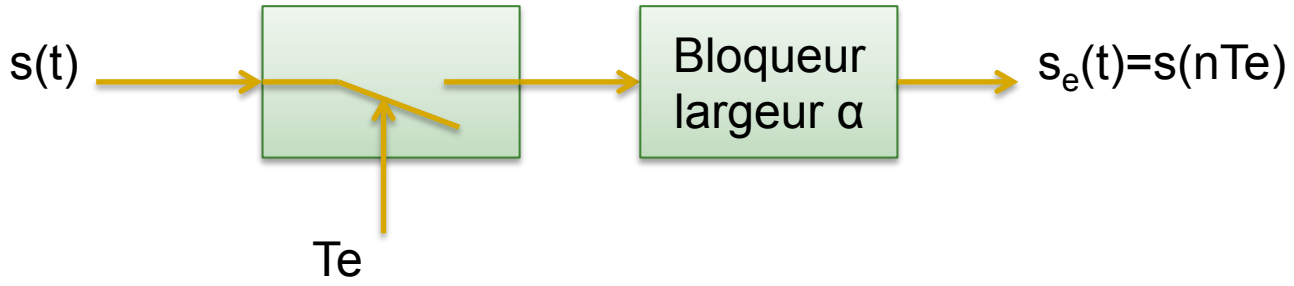
```
clear all;
close all;
clc;
Fs = 1000;  %fréquence d'échantillonnage
Ts = 1/Fs;  %période d'échantillonnage
L = 50*(Fs/100); %longueur de la séquence (50 périodes du signal)
t = Ts*(0:L-1); %vecteur temporel

% Signal d'entrée : sinusoïde de fréquence 100 Hz et d'amplitude 5volts
x = 5*sin(2*pi*100*t);

%Analyse spectrale
Y = fft(x); % transformée de fourier du
f = Fs*linspace(0,1,length(Y)); %abciss

figure(1);
plot(f,abs(Y))
title('spectre de X(f))')
xlabel('Frequences (Hz)')
ylabel('|X(f)|')
```



Echantillonnage réel

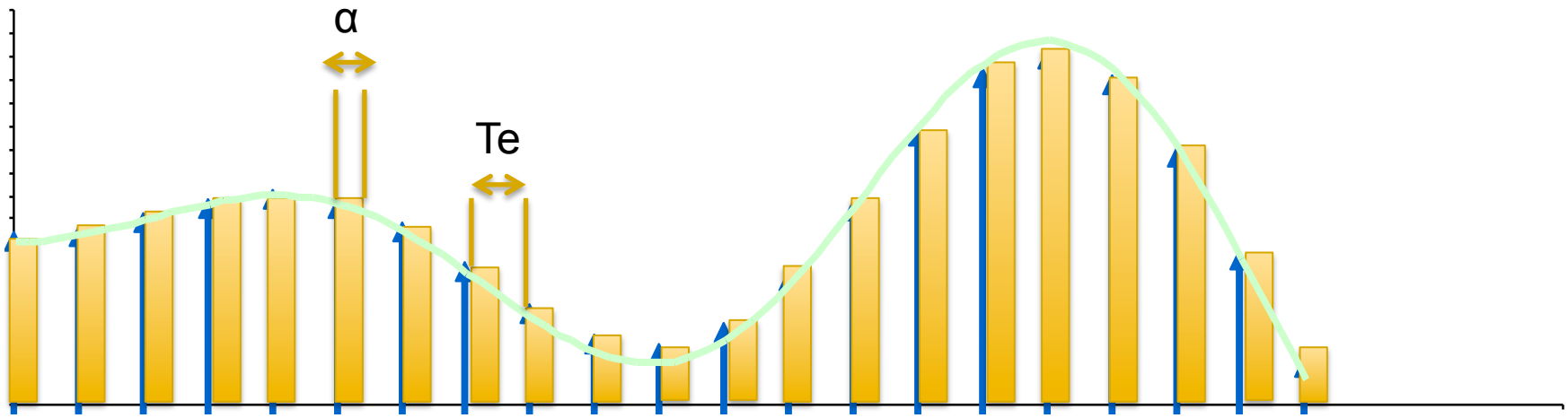
- L'échantillonnage idéal basé sur des impulsions ultra-courtes n'est pas réalisable. Dans la pratique, le dirac n'existe pas, les impulsions sont de durées finies α.
- Le signal échantillonné réel sera constitué d'une suite d'impulsion distantes de Te et de largeur α.
- La valeur du signal est généralement bloqué pendant α (bloqueur d'ordre 0).

Échantillonneur idéal = interrupteur



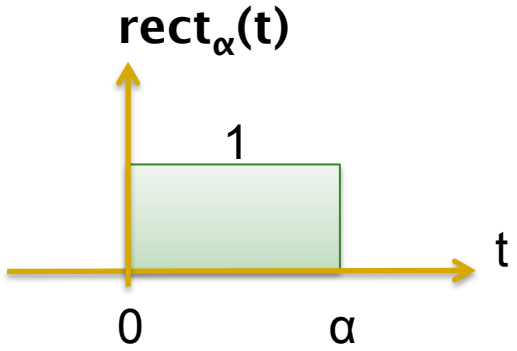
Echantillonnage réel

- L'allure temporelle du signal échantillonné est :



Echantillonnage réel

- L'échantillonnage réel correspond à multiplier le signal d'entrée par un peigne de porte de largeur α.



$$s_e(t) = s(t) \cdot \sum_{k=-\infty}^{+\infty} rect_\alpha(t - kT_e) = s(t).(Pgne_{T_e}(t) * rect_\alpha(t))$$

$$S_e(f) = S(f) * TF(\sum_{k=-\infty}^{+\infty} rect_\alpha(t - kT_e)) = S(f) * (\frac{1}{T_e} \sum_{k=-\infty}^{+\infty} \delta(f - k.f_e)).TF(rect_\alpha(t)))$$

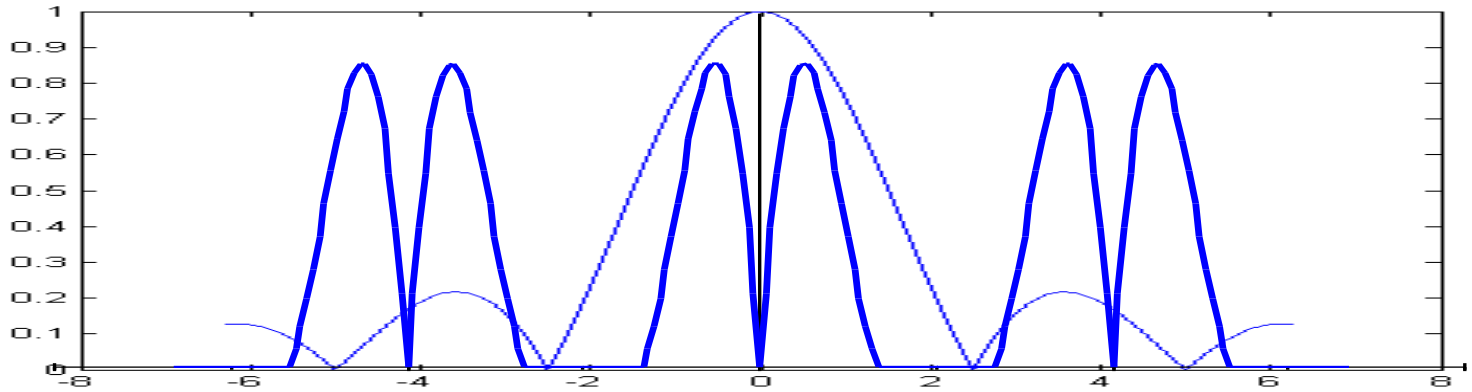
$$S_e(f) = \frac{1}{T_e} \sum_{k=-\infty}^{+\infty} S(f - k.f_e).TF(rect_\alpha(t)) = \left[\alpha.\frac{\sin(\pi.f.\alpha)}{\pi.f.\alpha}\right].\frac{1}{T_e} \sum_{k=-\infty}^{+\infty} S(f - k.f_e)$$

Echantillonnage réel

- Le spectre du signal échantillonné réel est :

$$S_e(f) = \frac{\alpha}{T_e} \sum_{k=-\infty}^{+\infty} \frac{\sin(\pi.k.f_e.\alpha)}{\pi.k.f_e.\alpha} .S(f - k.f_e)$$

- Le spectre correspond au spectre d'un signal échantillonné de manière idéal (réplication du spectre du signal d'entrée) mais modulé par un sinus cardinal.

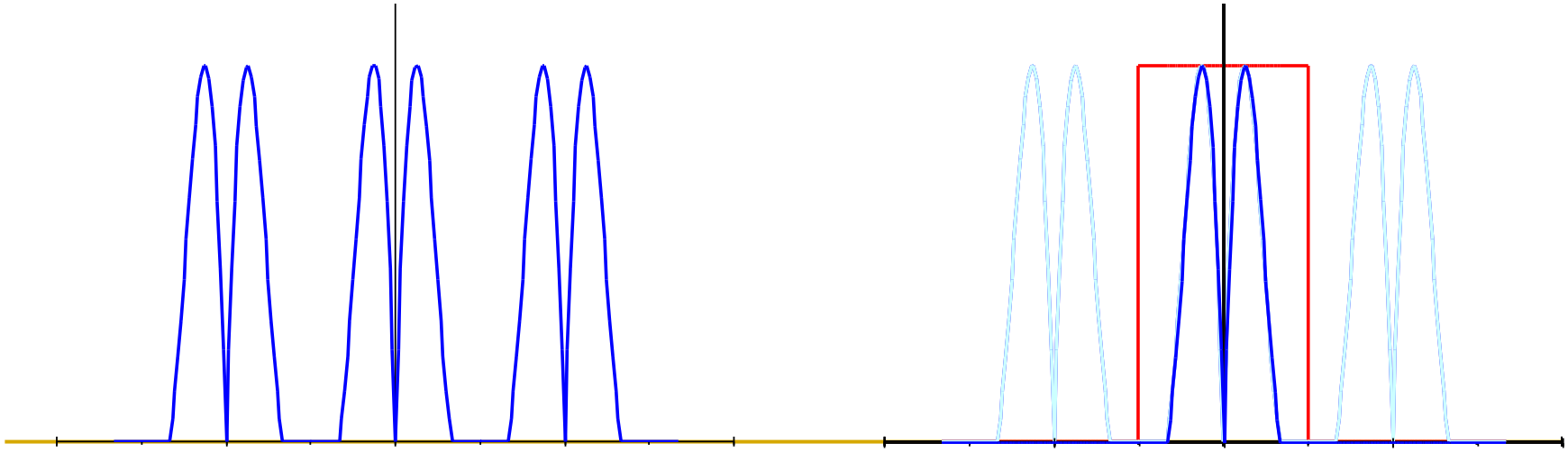


Reconstruction du signal original ?

- Si on dispose uniquement du signal échantillonné $S_e(t)$, peut on retrouver le signal original $S(t)$?
  - 1ère condition : le signal $S(t)$ est à spectre borné
  - 2ème condition : l'échantillonnage a été réalisé en respectant Shannon

- Filtre d'extraction du spectre de $S(f)$

$$S(f) = S_e(f) \cdot rect_{F_e}(f)$$



Reconstruction du signal original

- Le signal *s(t)* peut être obtenu par transformée de fourier inverse à partir du spectre du signal échantillonnée, filtré par un filtre passe-bas de largeur F_e

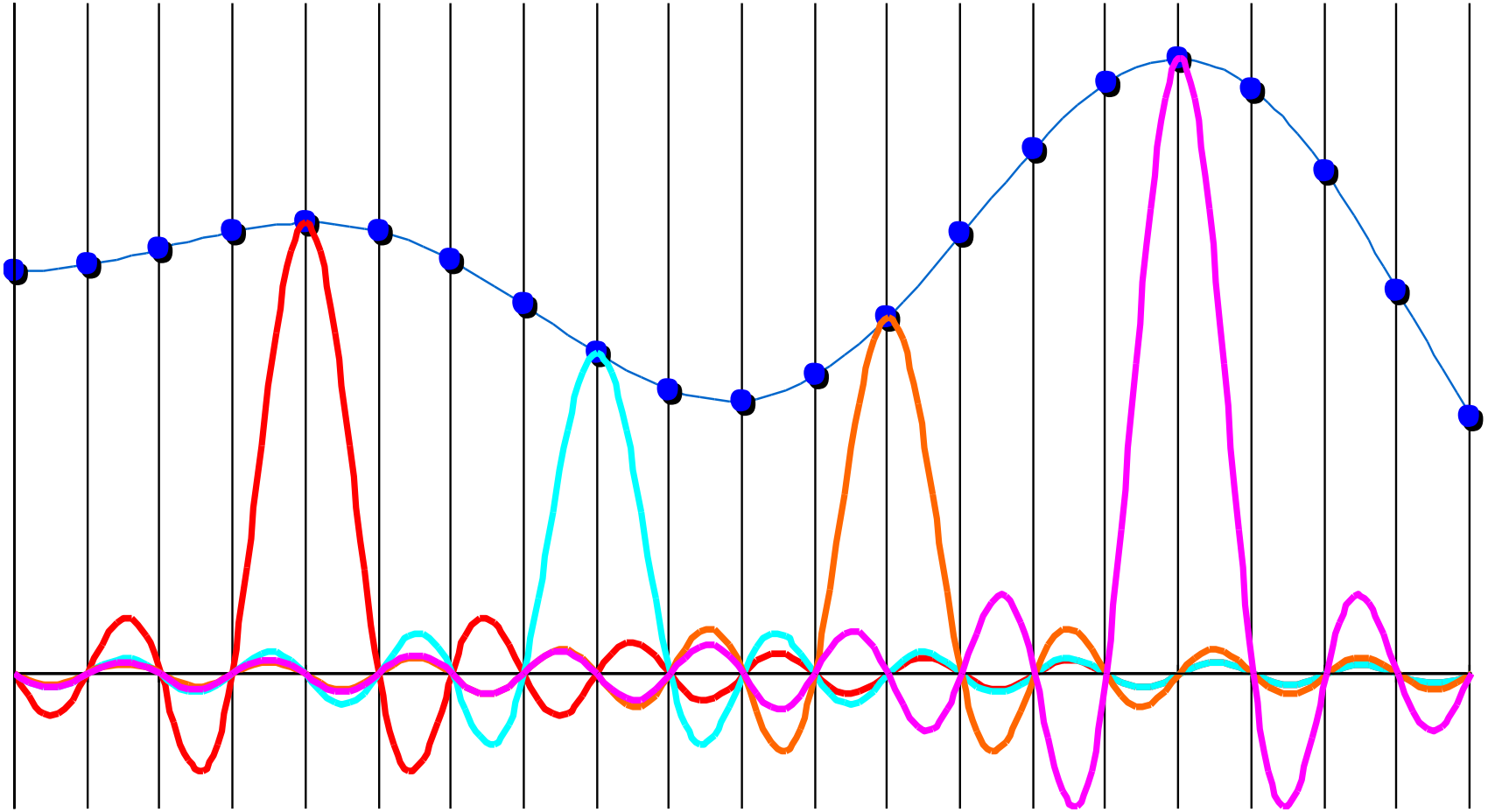
$$S(f) = S_e(f) \cdot rect_{F_e}(f)$$

TF^{-1}

$$s(t) = s_e(t) * f_e . \frac{1}{f_e} \frac{\sin(\pi . f_e . t)}{\pi . f_e . t} = \sum_{k=-\infty}^{+\infty} s(k.T_e).\delta(t - k.T_e) * \frac{\sin(\pi . f_e . t)}{\pi . f_e . t}$$

$$s(t) = \sum_{k=-\infty}^{+\infty} s(k.T_e). \frac{\sin(\pi . f_e . (t - k.T_e))}{\pi . f_e . (t - k.T_e)}$$

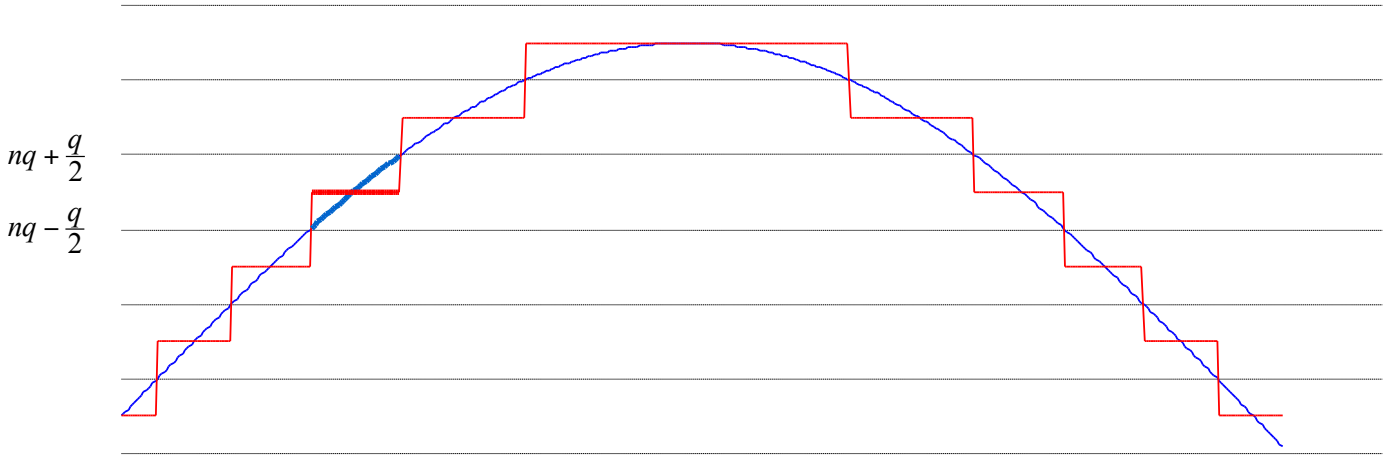
Reconstruction



Quantification

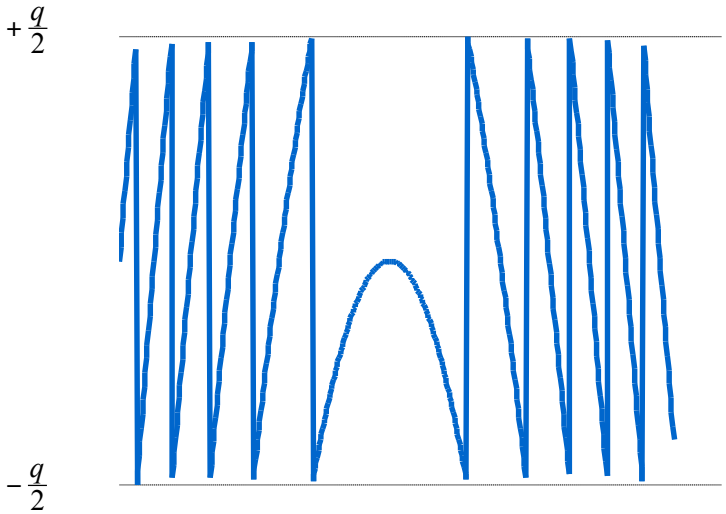
- Le signal échantillonné est quantifié en amplitude afin de fournir une représentation (format de codage) compatible avec les traitements numériques

$$s(nT) \longrightarrow mq$$

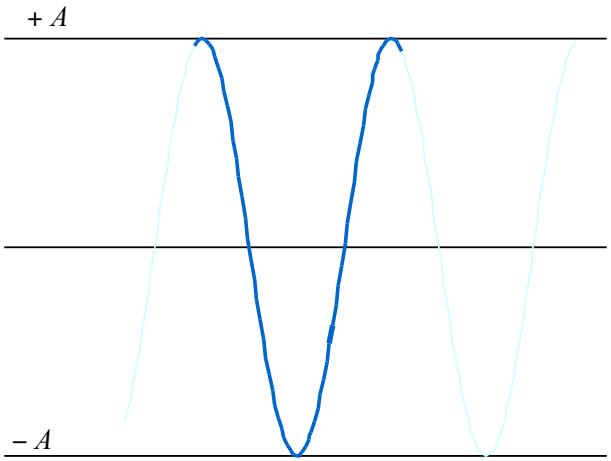


Bruit de quantification

- Le bruit de quantification est dû à l'erreur comise entre le signal réel et le signal discrétisé en amplitude.
- On montre que ce bruit à une valeur de $B = \frac{q^2}{12}$ ou q représente le quantum
- Le quantum est la plus petite valeur quantifiable ⇔ LSB



Quantification : dynamique de codage



- 2^N valeurs sur la dynamique totale du codeur soit:

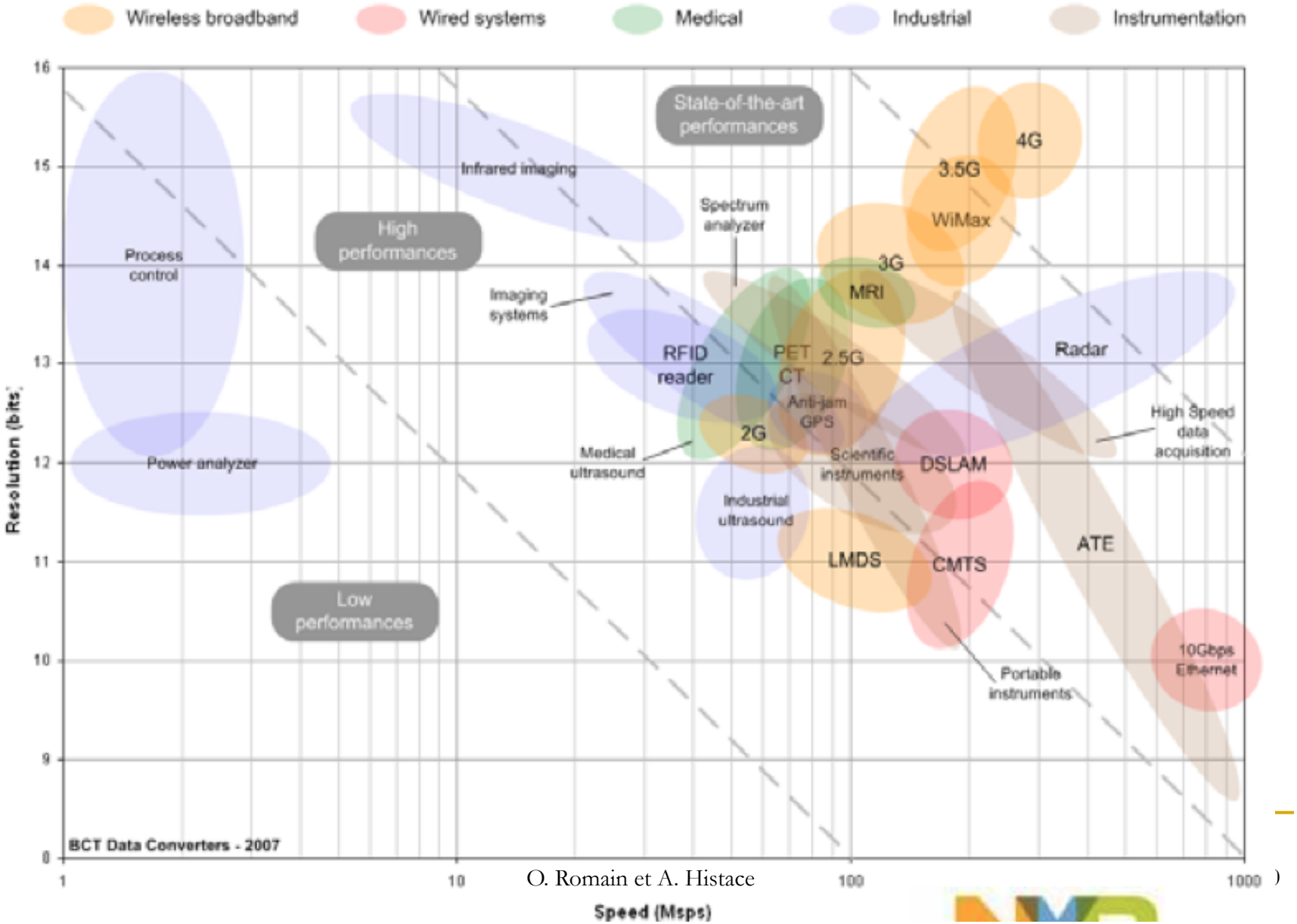
$$A = 2^{n-1} \cdot q$$

En exprimant la puissance du signal:

$$P_c = \frac{\left(2^{n-1} \cdot q\right)^2}{2} \qquad \frac{P_c}{B} = \frac{3}{2} \cdot 2^{2N}$$

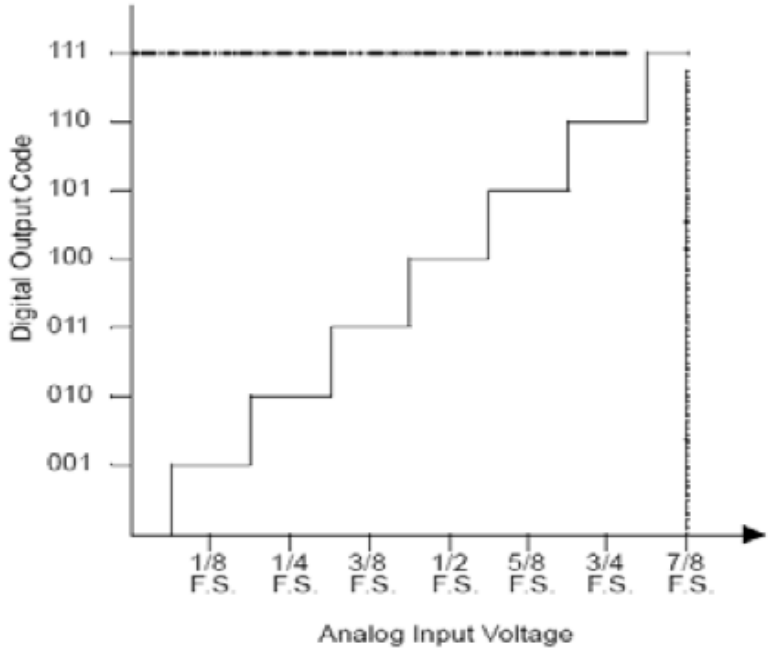
$$SNR = \left(\frac{P_c}{B}\right)_{dB} = 10 \cdot \log\left(\frac{3}{2} \cdot 2^{2N}\right) = 6.2 \cdot N + 1.76$$

CAN


Wireless broadband
Wired systems
Medical
Industrial
Instrumentation
State-of-the-art performances
High performances
Low performances
Infrared imaging
Spectrum analyzer
Process control
Imaging systems
Power analyzer
RFID reader
PET
CT
2.5G
Anti-jam GPS
2G
Medical ultrasound
Industrial ultrasound
Scientific instruments
DSLAM
LMDS
CMTS
ATE
Radar
High Speed data acquisition
Portable instruments
10Gbps Ethernet
4G
3.5G
WiMax
3G
MRI
BCT Data Converters - 2007
Resolution (bits)
Speed (Msps)
16
15
14
13
12
11
10
9
0
1
10
100
1000

CAN : caractéristique de transfert idéale

- Résolution : $x_{min}=q=x_{max}/(2^N-1)$
- N bits, 2^N états ; $12 bits \rightarrow 1\ bit = 0.0244\%$ FS
- Temps de conversion : Ts =1/Fs



Fonction de transfert d'un convertisseur analogique numérique

CAN : nombre de bits ?

RESOLUTION N	2^N	VOLTAGE (10 V FS)	ppm FS	% FS	dB FS
2-bit	4	2.5 V	250,000	25	–12
4-bit	16	625 mV	62,500	6.25	–24
6-bit	64	156 mV	15,625	1.56	–36
8-bit	256	39.1 mV	3,906	0.39	–48
10-bit	1,024	9.77 mV (10 mV)	977	0.098	–60
12-bit	4,096	2.44 mV	244	0.024	–72
14-bit	16,384	610 µV	61	0.0061	–84
16-bit	65,536	153 µV	15	0.0015	–96
18-bit	262,144	38 µV	4	0.0004	–108
20-bit	1,048,576	9.54 µV (10 µV)	1	0.0001	–120
22-bit	4,194,304	2.38 µV	0.24	0.000024	–132
24-bit	16,777,216	596 nV*	0.06	0.000006	–144

NOTES: *600 nV is the Johnson Noise in a 10 kHz BW of a 2.2 kΩ Resistor @ 25°C

10 bits and 10 V FS yields an LSB of 10 mV, 1000 ppm, or 0.1%.
All other values may be calculated by powers of 2.

Partie 3

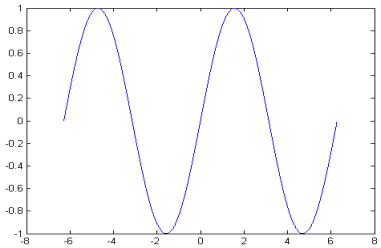
Analyse spectrale des signaux échantillonnés

Méthodes de calcul fréquentielles

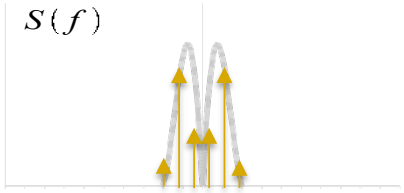
- L'analyse en fréquence des signaux permet de donner des informations supplémentaires à l'analyse en temps
- Les méthodes utilisées dépendent de la nature des signaux

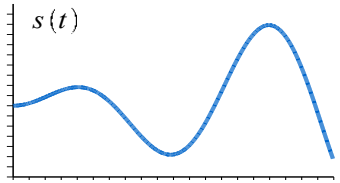
Signal	Spectre Méthode de calcul	Spectre caractéristiques
Continu périodique	Série de Fourier	Discret et périodique
Continu non périodique	Transformée de Fourier	Continu et périodique
Discret périodique	Transformée de Fourier Discrète	Discret et périodique
Discret non périodique	Transformée de Fourier	Continu et périodique

Méthodes de calcul fréquentielles

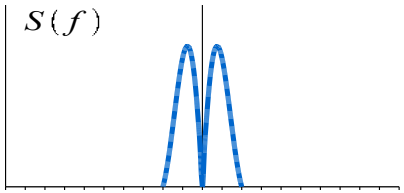


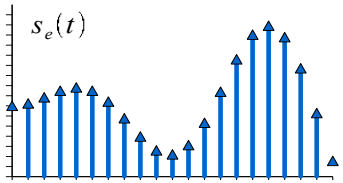
Série de Fourier



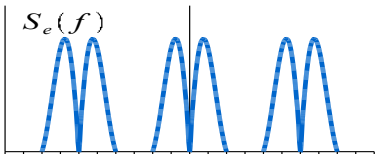


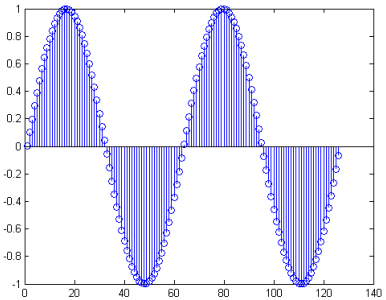
TF



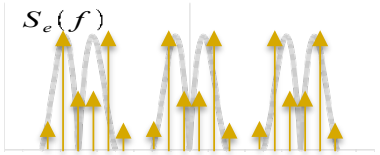


TF



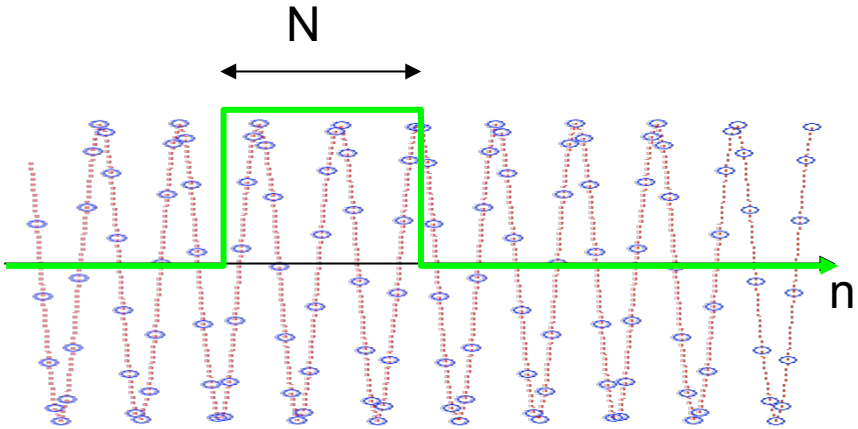


TFD



Transformée de Fourier Discrète : TFD

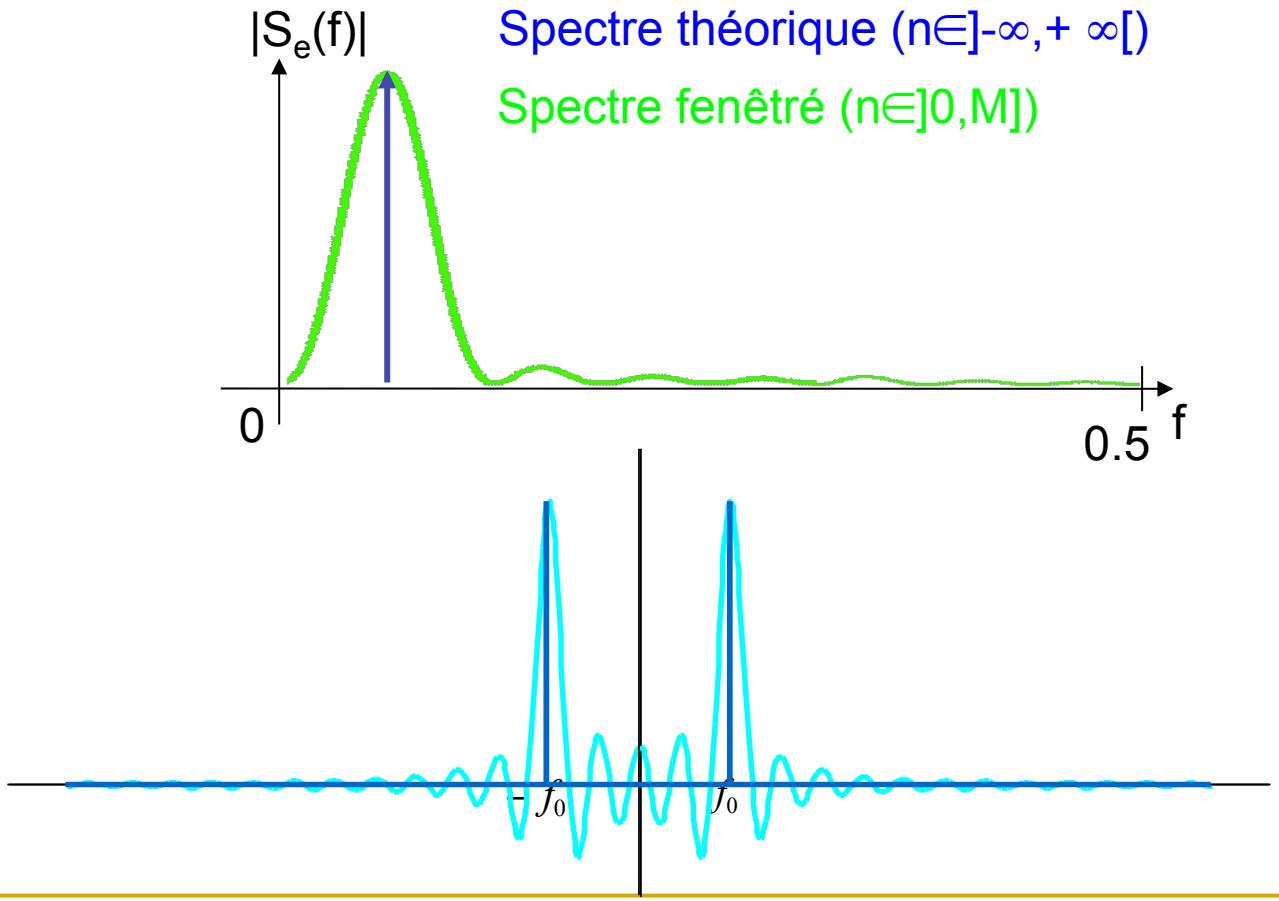
- Objectif : calculer la transformée de Fourier d'un signal *s(t)* à l'aide d'un calculateur.
- Contraintes matérielles (mémoire, temps de calcul, …) qui restreignent le nombre d'échantillon en entrée



$$s_e(t) = \sum_{k=0}^{N-1} s(k.Te) \cdot \delta(t - k.T_e).h(k.T_e)$$

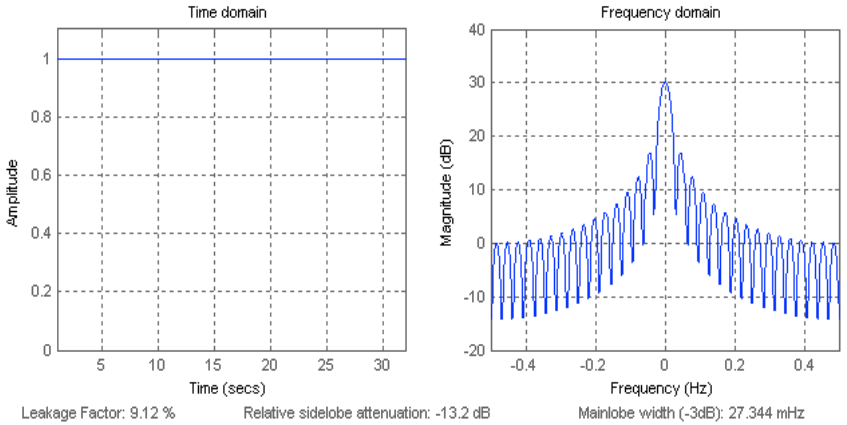
Signaux à durée limités

- Distorsion : effet de fenêtrage



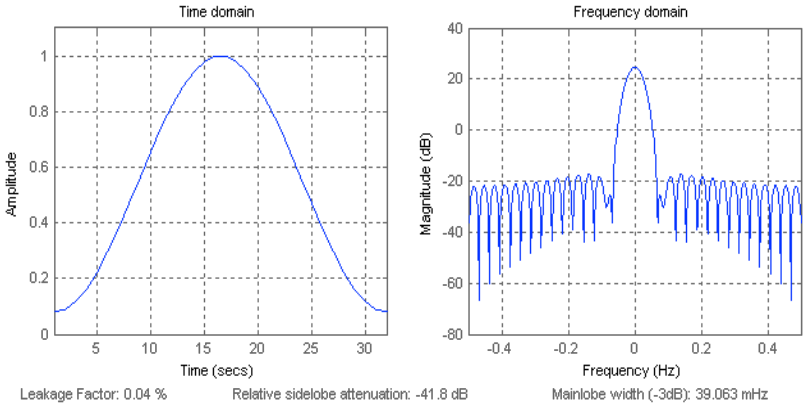
Fenêtrage

Rectangular



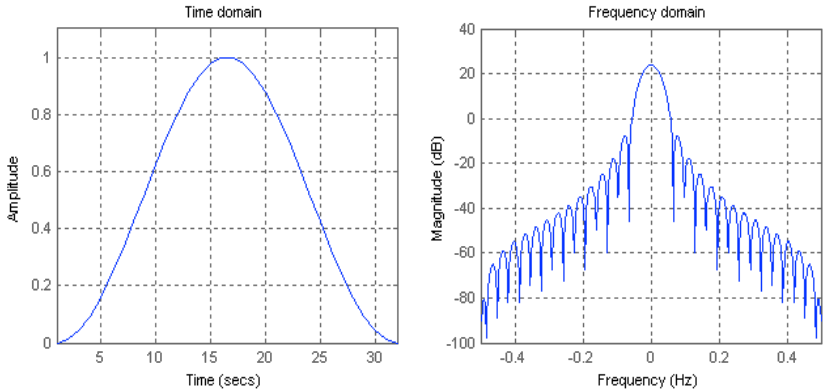
Leakage Factor: 9.12 % Relative sidelobe attenuation: -13.2 dB Mainlobe width (-3dB): 27.344 mHz

Hamming



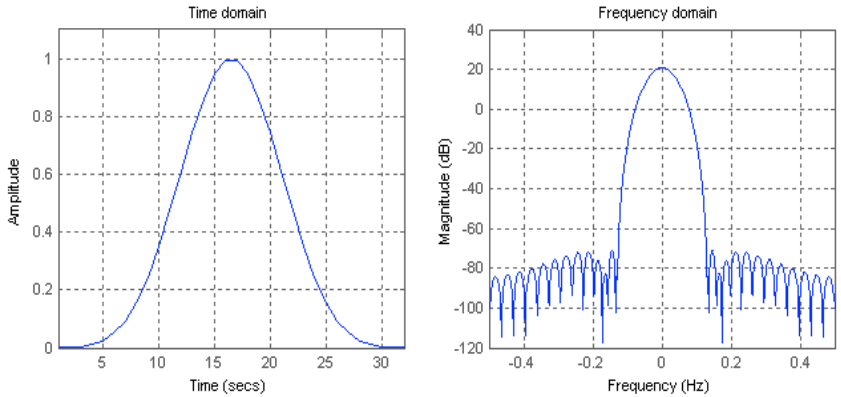
Leakage Factor: 0.04 % Relative sidelobe attenuation: -41.8 dB Mainlobe width (-3dB): 39.063 mHz

Hann



Leakage Factor: 0.05 % Relative sidelobe attenuation: -31.5 dB Mainlobe width (-3dB): 42.969 mHz

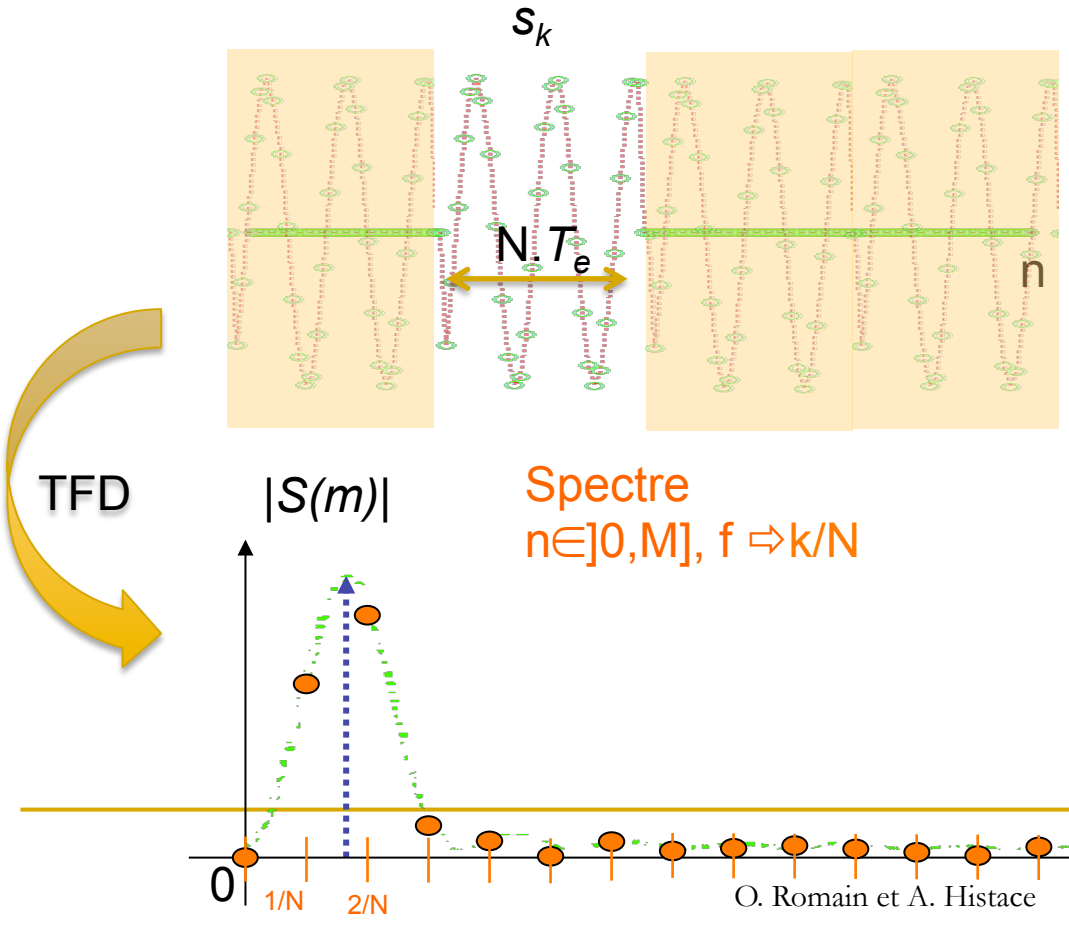
Blackmann-Harris



Leakage Factor: 0 % Relative sidelobe attenuation: -92 dB Mainlobe width (-3dB): 58.594 mHz

Transformée de Fourier Discrète : TFD

- Définition:
  - On appelle transformée de Fourier Discrète (TFD ou DFT) d'un signal défini sur par N échantillons, $s_{k,}$ la suite de N termes S_m, définie par :



$$S(m) = \sum_{k=0}^{N-1} s_k \cdot e^{-j.2.\pi.\frac{k.m}{N}}$$

Résumé TFD

- Soit *N* le nombre de points régulièrement espacés sur la période d'observation *T* .
- Le signal est défini par une suite de N valeurs s_k .
- Le spectre du signal est estimé par une suite de N valeurs S_k .

- Remarque : La TFD correspond à une Transformée de Fourier pour le signal fenêtré mais avec *f=1/(N.Te).*

- La transformée de Fourier Discrète Inverse correspond à :

$$s\ (k) = \frac{1}{N} . \sum_{k=0}^{N-1} S_m \cdot e^{j.2.\pi . \frac{k.m}{N}}$$

TFD et Matlab

fft

Discrete Fourier transform

Syntax

```
Y = fft(X)
Y = fft(X,n)
Y = fft(X,[],dim)
Y = fft(X,n,dim)
```

Definition

The functions `Y=fft(x)` and `y=ifft(X)` implement the transform and inverse transform pair given for vectors of length N by:

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)}$$

$$x(j) = (1/N)\sum_{k=1}^{N} X(k)\omega_N^{-(j-1)(k-1)}$$

where

$$\omega_N = e^{(-2\pi i)/N}$$

is an Nth root of unity.

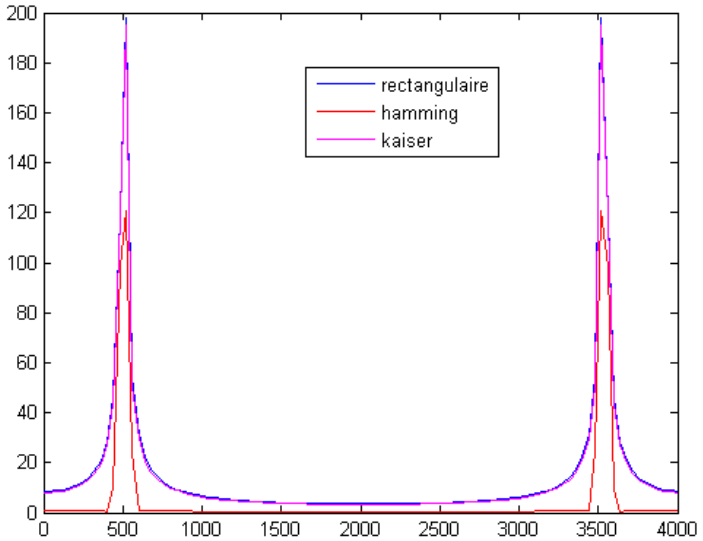
Matlab : TFD

```
clear all;
close all;
clc;
Fs = 1000;  %fréquence d'échantillonnage
Ts = 1/Fs;  %période d'échantillonnage
L = 500*(Fs/100); %longueur de la séquence (500 périodes du signal)
t = Ts*(0:L-1); %vecteur temporel

% Signal d'entrée : sinusoïde de fréquence 100 Hz et d'amplitude 5volts
x = 5*sin(2*pi*100*t);

%Analyse spectrale
Y = fft(x); % TFD du signal sur toute la longueur
f = Fs*linspace(0,1,length(Y)); %abcise des fréquences

figure(1);
plot(f,abs(Y))
title('spectre de X(f))')
xlabel('Frequences (Hz)')
ylabel('|X(f)|')
hold
line([500 500],[0 1400],'Color','r','LineWidth',1)

%Analyse spectrale sur une fenêtre de 200 échantillons
feq = Fs*linspace(0,1,length(x(100:300))); %abcise des fréquences
plot(feq,abs(fft(x(100:300))),'r')
```

Matlab : exercice

- Effet du fenêtrage sur la précision de la TFD

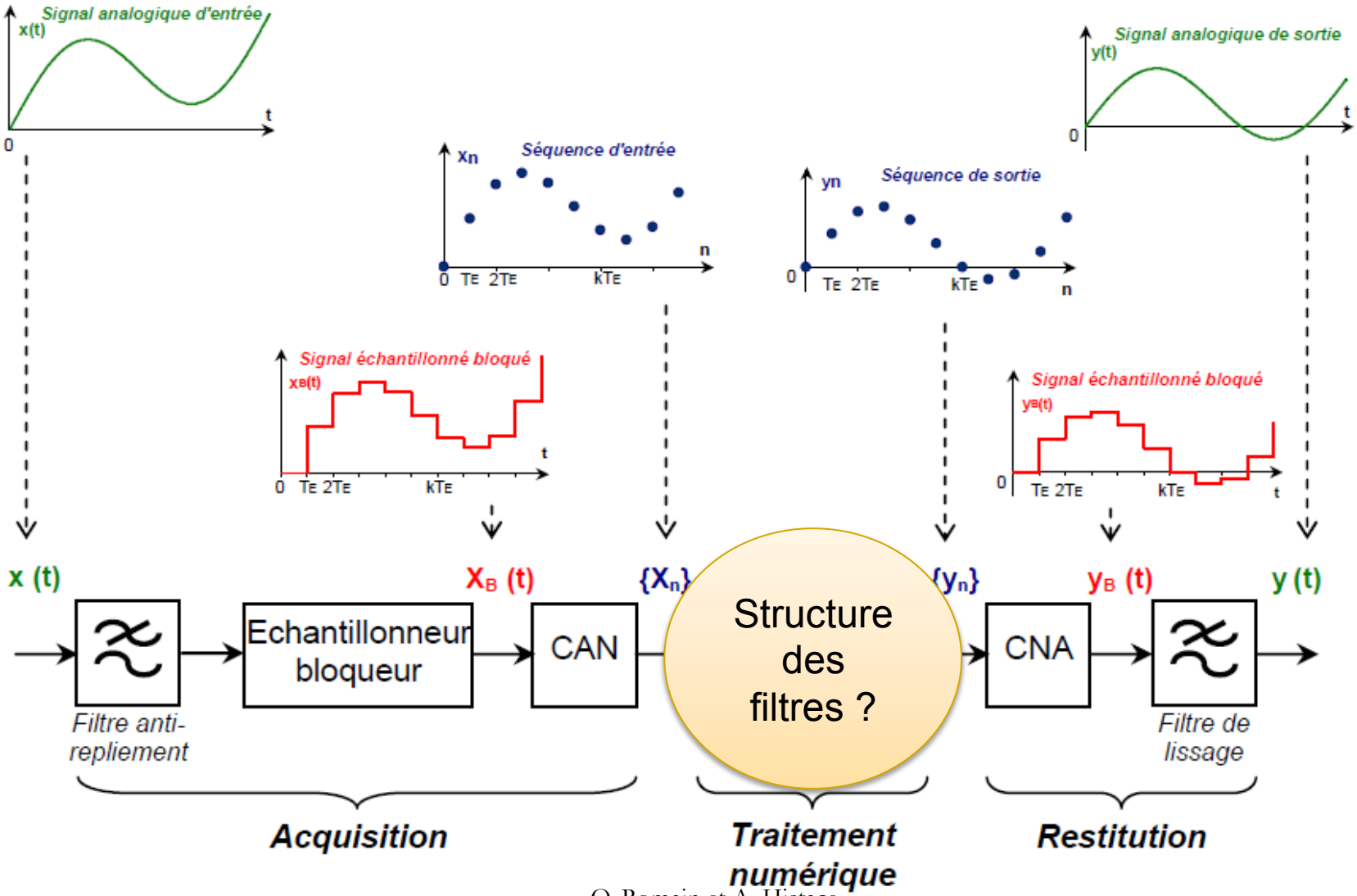
```
clear all;
close all;
clc;
Fs = 4000;  %fréquence d'échantillonnage
Ts = 1/Fs;  %période d'échantillonnage
L = 500*(Fs/100); %longueur de la séquence (500 périodes du signal)
t = Ts*(0:L-1); %vecteur temporel

% Signal d'entrée : sinusoïde de fréquence 500 Hz et d'amplitude 5volts
x = 5*sin(2*pi*500*t);
se = x(100:200);
te = t(100:200);
figure(1);
plot(t,x);
hold
plot(te,se,'.m');
axis([t(50) t(400) -6 6]);

%Analyse spectrale sur une fenêtre de 201 échantillons
figure(3);
Xw=fft(se);
fw=Fs*linspace(0,1,length(Xw));
plot(fw,abs(Xw));
hold

win=hamming(length(se))'; %utilisation d'une fenêtre de hamming
Xwham=fft(win.*se);
fwham=Fs*linspace(0,1,length(Xwham));
plot(fw,abs(Xwham),'r');

win=kaiser(length(se))'; %utilisation d'une fenêtre de blackmann
Xkaiser=fft(win.*se);
fkaiser=Fs*linspace(0,1,length(Xkaiser));
plot(fkaiser,abs(Xkaiser),'m');
legend('rectangulaire','hamming','kaiser');
```



Partie 3

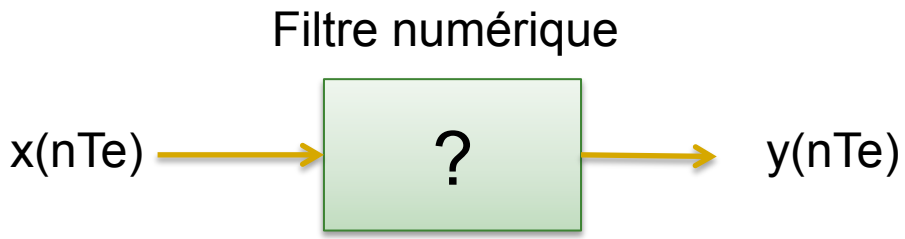
Filtres FIR et RII
Structures et synthèses

Structure d'une TNS



Introduction

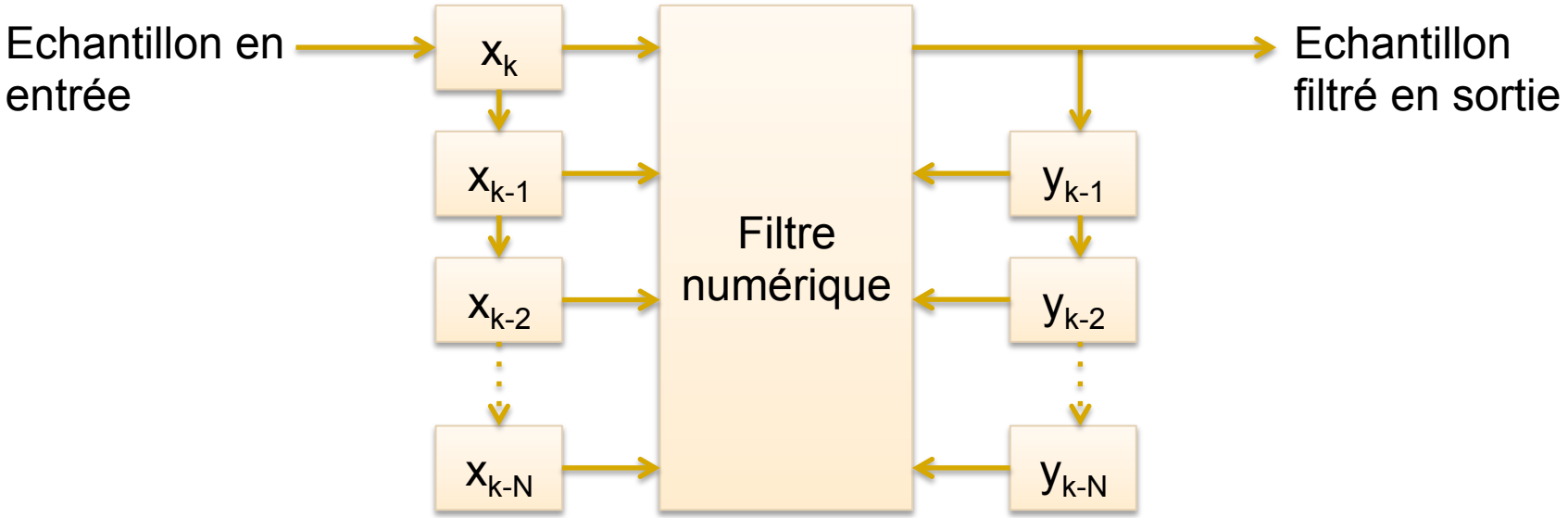
- Définition
  - On appelle « filtre numérique » un système utilisé pour modifier les caractéristiques fréquentielles d'un signal numérique en entrée selon un gabarit donné.
  - Un filtre numérique transforme un signal discret (entrée) en un autre signal discret (sortie)



  - Filtres numériques sont pour le discret ce que les filtres analogiques sont pour le continu
- Remarque :
  - Avec les circuits numériques de traitement (DSP, FPGA, GPU, etc.), les filtres numériques sont devenus plus intéressants du fait de la précision, fiabilité, stabilité, adaptabilité, etc.

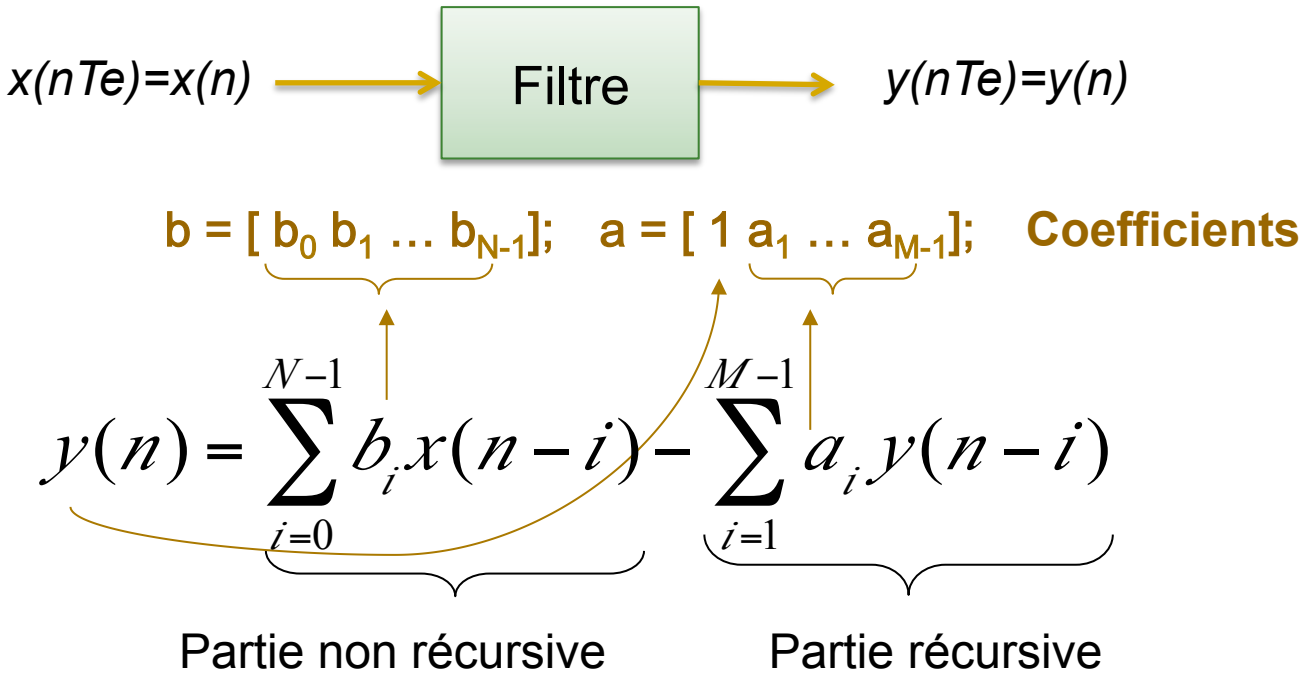
Filtres numériques

- Comme pour les filtres analogiques, le problème consiste à réaliser un filtre donnant une réponse fréquentielle *H(f)* donnée, une réponse impulsionnelle *h(t)* fixée ou éventuellement une réponse indicielle.
- D'une manière générale, la sortie y_k est une fonction de (x_k, x_{k-1},…, x_{k-N}, y_{k-1}, y_{k-2}, …,y_{k-N}).



Filtre numérique

- La fonction générale de filtrage numérique (quelque soit sa structure), c'est à dire de calcul de l'échantillon y_k de sortie, répond à l'équation générale dite **« de différence »** :

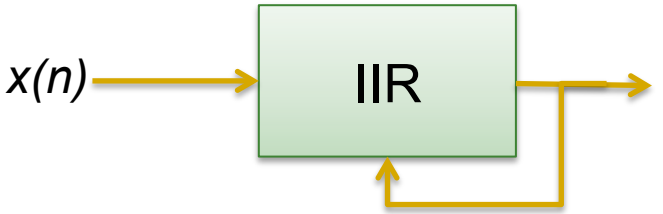


Filtres numériques

- Filtres non-récursifs : FIR – Filtre à Réponse Impulsionnelle Finie

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i)$$

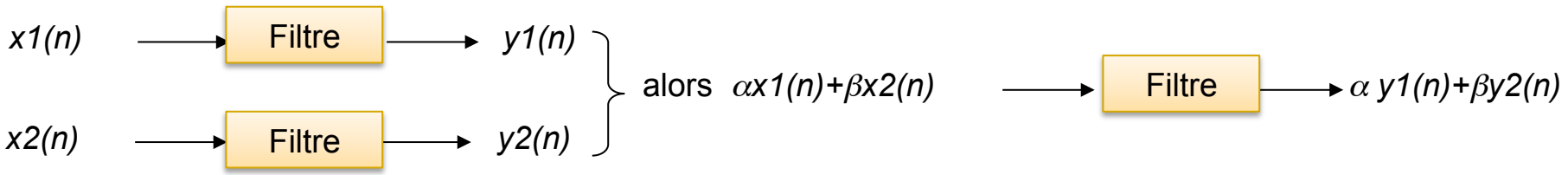
- Filtres récursifs : IIR – Filtre à Réponse Impulsionnelle Infinie



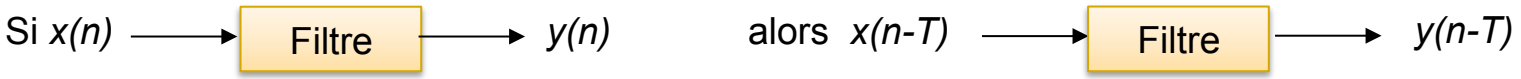
$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=1}^{M-1} a_i y(n-i)$$

Filtres numériques : propriétés

- Linéarité

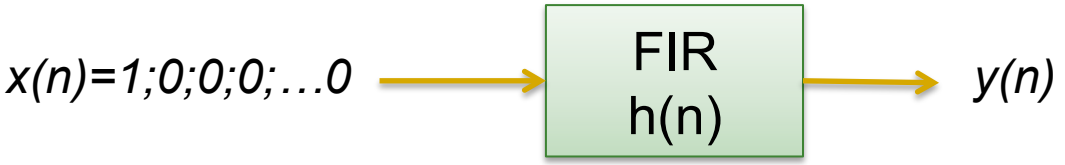


- Retard



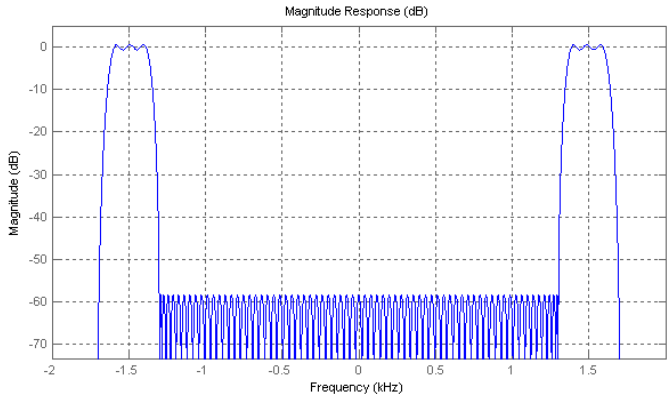
Réponse impulsionnelle

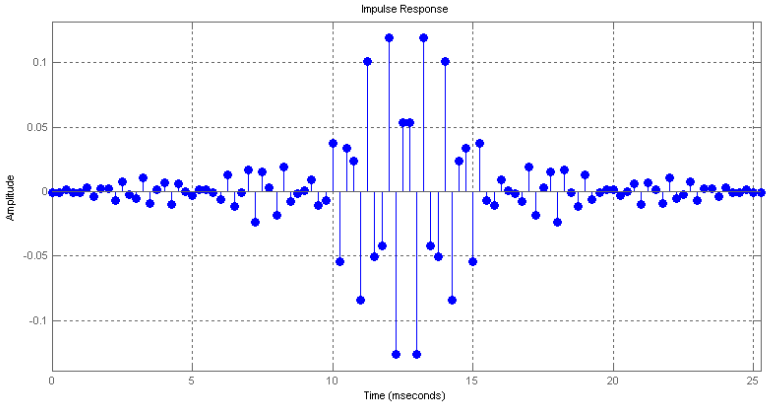
- Filtre FIR
  - La réponse impulsionnelle est obtenue sur *y(n)* lorsque *x(n)* est un dirac à t=0
    - *x(0) = 1; x(1) = 0; x(2) = 0; x(3) = 0; …….. x(N) = 0*



$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i)$$

$$h(k) = b(k)$$



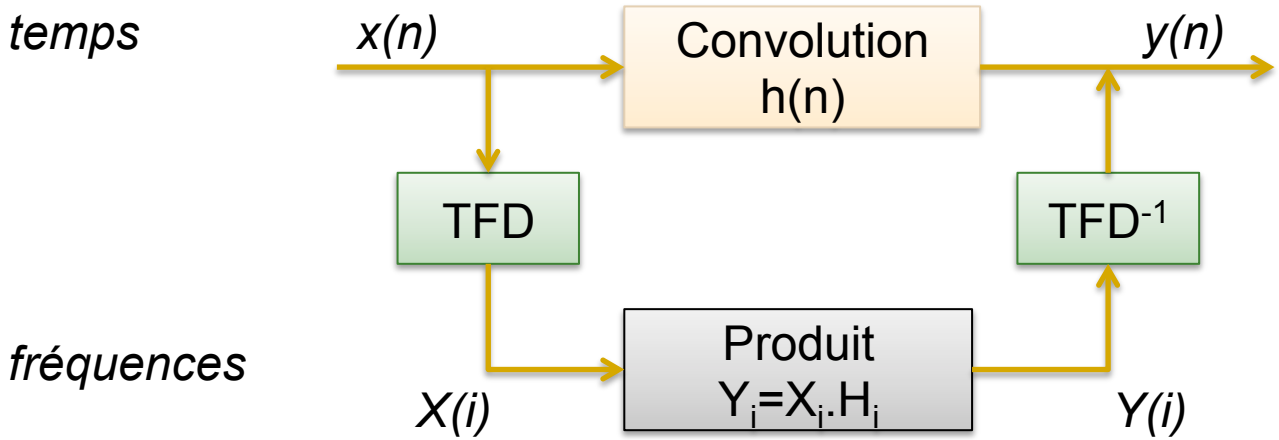


Réponse impulsionnelle

- Filtre FIR
  - *x(0) = 1; y(0) = x(0).h(n-1) = h(n-1)*
  - *x(1) = 0; y(1) = x(0).h(n-2) + x(1).h(n-1) = h(n-2)*
  - *x(2) = 0; y(2) = x(0).h(n-3) + x(1).h(n-2) + x(2).h(n-1) = h(n-3)*
  - *x(3) = 0; y(3) = x(0).h(n-4) + x(1).h(n-3) + x(2).h(n-2) + x(3).h(n-1) = h(n-4)*
  - *…*
  - *…*

Filtres numériques

- Remarques
  - Comme pour les filtres analogiques le filtrage peut être réalisé dans l'espace des fréquences via un produit du spectre du signal échantillonnée par la réponse fréquentielle du filtre



- 3 phases pour la conception des filtres numériques
  - Détermination des coefficients de h(n)
  - Synthèse du filtre par l'équation aux différences
  - Programmation du filtre (algorithme) sur la cible matérielle

Transformée en Z

- La transformée en Z est l'équivalent de la transformée de Laplace pour les signaux continus

- La transformée en Z d'un signal s(t) échantillonnée à Te est donnée par :

$$S(z) = \sum_{k=0}^{\infty} s(k.T_e).z^{-k} = \sum_{k=0}^{\infty} s(k.T_e).e^{-k.p.T_e}$$

- L'origine de la transformée en Z provient du calcul de la transformée de Laplace du signal échantillonné

Transformée en Z

- Démonstration :

$$s_e(t) = s(t) \cdot \sum_{k=-\infty}^{+\infty} \delta(t - kT_e) = s(t).Pgne_{T_e}(t) = \sum_{k=-\infty}^{+\infty} s(k.T_e).\delta(t - kT_e)$$

$$S_e(p) = \int_0^{\infty} s_e(t).e^{-p.t}.dt = \int_0^{\infty} \sum_{k=-\infty}^{+\infty} s(k.T_e).\delta(t - kT_e).e^{-p.t}.dt$$

$$S_e(p) = \sum_{k=-\infty}^{+\infty} s(k.T_e).\int_0^{\infty} .\delta(t - kT_e).e^{-p.t}.dt = \sum_{k=-\infty}^{+\infty} s(k.T_e).e^{-p.k.T_e}$$

$$z = e^{p.T_e} \longrightarrow S_e(z) = \sum_{k=-\infty}^{+\infty} s(k.T_e).z^{-k}$$

CQFD

Transformée en Z : propriétés

- Théorème du retard

$$w(n) = x(n-m) \xrightarrow{TFZ} W(z) = X(Z).z^{-m}$$

- Opérateur retard unité
  - L'opérateur retard fait correspondre à un signal le même signal mais retardé d'un échantillon

- Linéarité

$$w(n) = x(n) + y(n) \xrightarrow{TFZ} W(z) = \sum_{k=-\infty}^{+\infty} (x(k.T_e) + y(k.T_e).z^{-k} = X(z) + Y(z)$$

Transformée en Z : propriétés

- Convolution

$$w(n) = x(n) * y(n) \xrightarrow{TFZ} W(Z) = X(Z).Y(Z)$$

- Multiplication par une exponentielle

$$w(n) = x(n).a^n \xrightarrow{TFZ} W(Z) = X(\frac{Z}{a})$$

- Linéarité

$$w(n) = x(n).n \xrightarrow{TFZ} W(Z) = -Z.\frac{dX(Z)}{dZ}$$

Transformée en Z inverse

- La transformée en Z inverse permet de retrouver le signal échantillonnée temporellement

$$X(Z) \xrightarrow{TFZ^{-1}} x(n)$$

- Expression

$$x(n) = \frac{1}{2.\pi.j} \oint X(Z).Z^{n-1}.dZ$$

- Théorème des résidus

Exemple filtre numérique

- Filtre réalisant une moyenne sur deux échantillons consécutifs :

$$y(n) = \frac{x(n) + x(n-1)}{2} \xrightarrow{TFZ} Y(z) = \frac{X(Z) + Z^{-1}.X(Z)}{2} = X(Z).(\frac{1+Z^{-1}}{2})$$

$$H(Z) = \frac{1+Z^{-1}}{2}$$

$$h(0) = 0.5$$

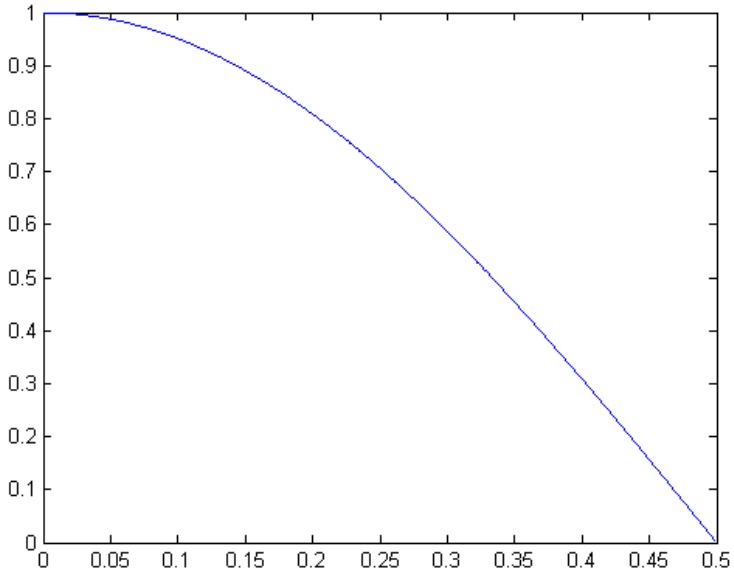
$$h(1) = 0.5$$

$$H(p) = \frac{1+e^{-p.T_e}}{2} = e^{\frac{-p.T_e}{2}}.\frac{(e^{\frac{p.T_e}{2}} + e^{\frac{-p.T_e}{2}})}{2}$$

$$H(f) = \cos(\pi.f.T_e).e^{-j.\pi.f.T_e}$$

Exemple filtre numérique

```
clear all;
close all;
clc;

numerateur=[0.5 0.5];
denominateur=1;
sys=tf(numerateur,
denominateur, 1);

[H,W] = freqz([0.5 0.5],
1,100); %réponse fréquentielle
du filtre, en Z
plot(W/(2*pi),abs(H))
%affichage en fréquence
```



Fonction de transfert

- Equation aux différences

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=1}^{M-1} a_i y(n-i)$$

TFZ

$$Y(Z) = X(Z).\sum_{i=0}^{N-1} b_i.Z^{-i} - Y(Z).\sum_{k=1}^{M-1} a_i.Z^{-k}$$

H(Z)

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{\sum_{i=0}^{N-1} b_i.Z^{-i}}{\sum_{k=0}^{M-1} a_i.Z^{-k}}$$

Fonction de transfert

- FIR :
  - Filtres non récursifs

$$H(Z) = \frac{Y(Z)}{X(Z)} = \sum_{i=0}^{N-1} b_i . Z^{-i}$$

- RII
  - Filtres récursifs

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{\sum_{i=0}^{N-1} b_i . Z^{-i}}{\sum_{k=0}^{M-1} a_i . Z^{-k}}$$

Synthèse des filtres

- Transformation de H(p) en H(z)
  - La méthode la plus utilisée pour déterminer la fonction de transfert d'un filtre numérique, consiste à transposer la fonction de transfert H(p) dans le plan Z par une règle de transformation de p->Z

$$z = e^{p.T_e} \longrightarrow p = \frac{\ln(z)}{T_e}$$

  - Pas envisageable car H(z) ne sera pas un polynôme en Z
    - Ne correspond pas à l'équation aux différences

$$H(p) = \frac{1}{p^2 + p + 1} \longrightarrow H(z) = \frac{T_e^2}{\ln(Z)^2 + T_e.\ln(z) + T_e^2}$$

Synthèse des filtres

- Objectif : Trouver une transformation qui conserve le « quotient de deux polynômes »
  - Contraintes matérielles

- Les principales méthodes de synthèse sont :
  A. Méthode de l'invariance impulsionnelle
  B. Méthode de l'invariance indicielle
  C. Transformation d'Euler ou équivalence de la dérivation
  D. Transformation bilinéaire

A. Méthode de l'invariance impulsionnelle

Méthode de l'invariance impulsionnelle

- Objectif :
  - Réponse impulsionnelle du filtre numérique identique à la réponse impulsionnelle du filtre analogique échantillonnée.

$$H_e(f) = TF(h(t)) * TF(Pgn_{Te}(t)) = H(f) * \sum_{k=-\infty}^{+\infty} \delta(f - k.f_e)$$

$$h_e(t) = T_e.h(t).\sum_{k=-\infty}^{+\infty} \delta(T - k.T_e) = T_e.\sum_{k=-\infty}^{+\infty} h(k.T_e).\delta(T - k.T_e)$$

$$H_e(z) = T_e.\sum_{k=-\infty}^{+\infty} h(k.T_e).z^{-k}$$

Exemple : filtre passe-bas : FIR

- Filtre passe-bas
  - Conditions :
    - fréquence de coupure <<Fe
    - Filtres passe-bas ou filtre passe-bande

$$H(p) = \frac{1}{a+p} \longrightarrow h(t) = u(t).e^{-a.t}$$

$$h_e(t) = T_e.\sum_{k=0}^{+\infty} u(k.T_e).e^{-a.k.T_e}.\delta(T - k.T_e)$$

$$H_e(z) = T_e.\sum_{k=0}^{+\infty} e^{-a.k.T_e} z^{-k} = T_e.\sum_{k=0}^{+\infty} (e^{-a.T_e} z^{-1})^k = \frac{T_e}{1 - e^{-a.T_e}.z^{-1}} = \frac{z.T_e}{z - e^{-a.T_e}}$$

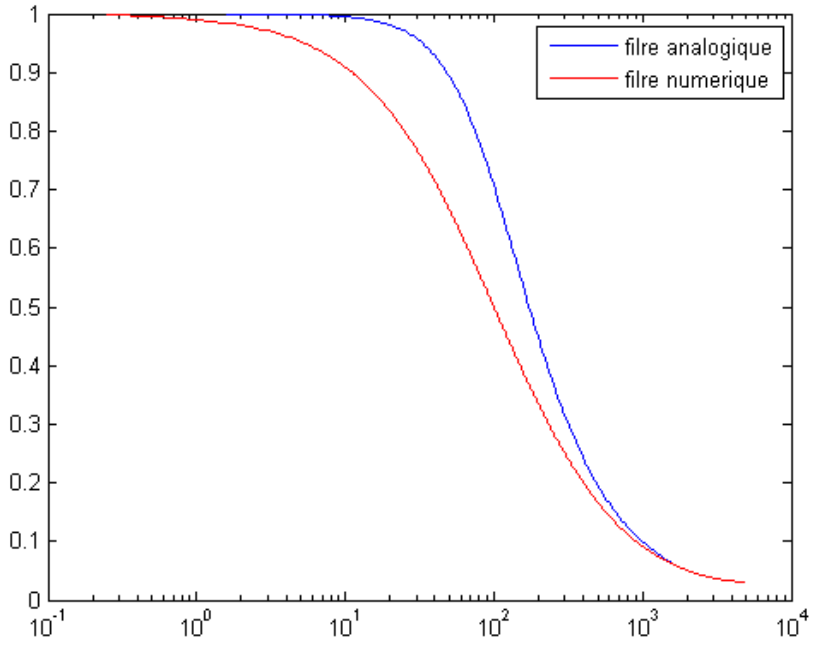
Exemple : filtre passe-bas Matlab

- Application : Filtre passe-bas 100Hz avec Fe=10kHz

```
clear all;
close all;
clc;

fo=100; %100Hz
fe=10000; %10kHz pour la frÈquence d'Èchantillonnage


%filtre analogique
num=[1];
den=[1 2*fo*pi];
[H,W] = freqs(num,den); %rÈponse frÈquentielle du filtre,
en Z
semilogx(W/(2*pi),abs(H)/max(abs(H))) %affichage en
frÈquence
hold

%filtre numÈrique
numerateur=[1/fe 0];
denominateur=[1 -exp(-j*2*fo*pi/fe)];


sys=tf(numerateur, denominateur, fe);
[H1,W1] = freqz(numerateur,denominateur,20000); %rÈponse
frÈquentielle du filtre, en Z
semilogx(W1/(2*pi)*fe,(abs(H1)/max(abs(H1))),'r')
%affichage en frÈquence
legend('filre analogique','filre numerique');
```

$$a = 2.\pi.f_0.j = 200\pi.j$$

$$H_e(z) = \frac{0.0001.z}{z - e^{-0.02.\pi.j}}$$



B. Méthode de l'invariance indicielle

Méthode de l'invariance indicielle

- Objectif :
  - Réponse indicielle du filtre numérique identique à la réponse indicielle du filtre analogique échantillonnée.

$$H(p) = \frac{1}{1+\tau p} \longrightarrow Y(p) = \frac{1}{p}.\frac{1}{1+\tau p} \longrightarrow y(t) = (1-e^{-\tau.t}).u(t)$$

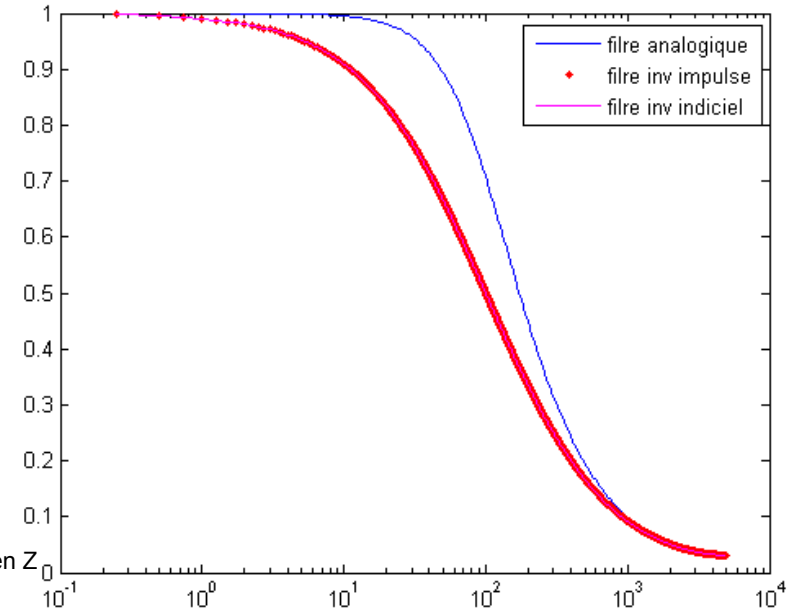
$$y(n) = (1-e^{-\tau.n.T_e}).u(n) \xrightarrow{TFZ} Y(Z) = \frac{1}{1-z^{-1}} - \frac{1}{1-e^{-\frac{T_e}{\tau}}.z^{-1}} = H(z).\frac{1}{1-z^{-1}}$$

$$H(z) = \frac{(1-e^{-\frac{T_e}{\tau}}).z^{-1}}{1-e^{-\frac{T_e}{\tau}}z^{-1}}$$

Exemple : filtre passe-bas Matlab

- Application : Filtre passe-bas 100Hz avec Fe=10kHz invariance indicielle

```
clear all;
close all;
clc;

fo=100; %100Hz
fe=10000; %10kHz pour la frÈquence d'Èchantillonnage

%filtre analogique
num=[1];
den=[1 2*fo*pi];
[H,W] = freqs(num,den); %rÈponse frÈquentielle du filtre, en Z
semilogx(W/(2*pi),abs(H)/max(abs(H))) %affichage en frÈquence
hold

%filtre numÈrique invariance impulsionnelle
num_impulse=[1/fe 0];
denom_impulse=[1 -exp(-j*2*fo*pi/fe)];

sys=tf(num_impulse, denom_impulse, fe);
[H1,W1] = freqz(num_impulse,denom_impulse,20000); %rÈponse frÈquentielle du filtre, en Z
semilogx(W1/(2*pi)*fe,(abs(H1)/max(abs(H1))),'.r') %affichage en frÈquence
legend('filre analogique','filre numerique');

% filtre numÈrique invariance indicielle
num_indiciel=[0 (1-exp(-j*2*fo*pi/fe))];
denom_indiciel=[1 -exp(-j*2*fo*pi/fe)];

sys2=tf(num_impulse, denom_impulse, fe);
[H2,W2] = freqz(num_indiciel,denom_indiciel,20000); %rÈponse frÈquentielle du filtre, en Z
semilogx(W2/(2*pi)*fe,(abs(H2)/max(abs(H2))),'m') %affichage en frÈquence
legend('filre analogique','filre inv impulse','filre inv indiciel');
```



C. Méthode d'Euler

Approximation de la dérivée

Enoncé du problème

- Une fonction de transfert analogique à la forme générale suivante :

$$H(p) = \frac{a_0 + a_1.p + a_2.p^2 + ... + a_n.p^n}{1 + b_1.p + b_2.p^2 + ... + b_m.p^m}$$

- La réponse temporelle répond à l'équation différentielle

$$y(t) + b_1.\frac{dy}{dt} + b_2.\frac{d^2y}{dt^2} + ... + b_m.\frac{d^my}{dt^m} = a_0.x(t) + a_1.\frac{dx}{dt} + a_2.\frac{d^2x}{dt^2} + ... + a_n.\frac{d^mx}{dt^m}$$

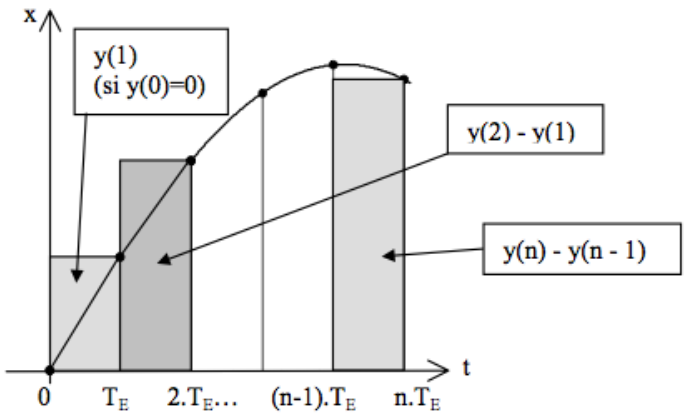
- Approximation de la dérivée
  - Si Te<<1

$$y(t) = \frac{dx}{dt} \xrightarrow{Te<<1} y(n) = \frac{x(n) - x(n-1)}{T_e} \xrightarrow{TFZ} Y(z) = \frac{1 - z^{-1}}{T_e}.X(z)$$

Enoncé du problème

- Approximation de la dérivée
  - Si Te<<1

$$y(t)=\frac{dx}{dt} \xrightarrow{Te<<1} y(n)=\frac{x(n)-x(n-1)}{T_e} \xrightarrow{TFZ} Y(z)=\frac{1-z^{-1}}{T_e}.X(z)$$



$$Y(p)=\frac{X(p)}{p} \xrightarrow{Euler} Y(z)=\frac{X(z)}{\frac{1-z^{-1}}{T_e}}$$

$$Y(z)=\frac{X(z)}{\frac{1-z^{-1}}{T_e}} \xrightarrow{TFZ^{-1}} \frac{y(n)-y(n-1)}{T_e}=x(n)$$

Enoncé du problème

$$y(t) = \frac{dx}{dt} \longrightarrow Y(p) = p.X(p) \xrightarrow{TFZ} Y(z) = \frac{1 - z^{-1}}{T_e}.X(z)$$

- Transformation

$$p \xrightarrow{Transformation} \frac{1 - z^{-1}}{T_e}$$

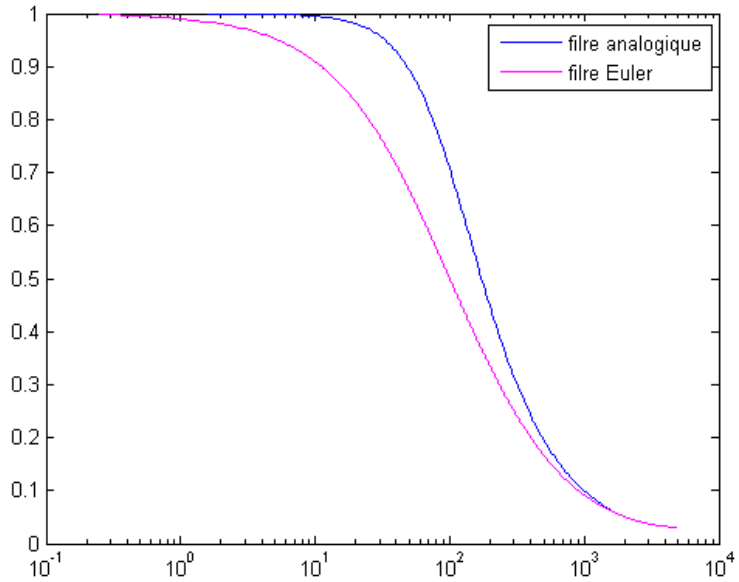
- Méthode
  - Détermination de la fonction de transfert souhaitée dans le domaine continu : H(p)
  - Remplacer p par $(1\text{-}z^{-1})/T_e$ pour trouver la fonction H(z)

$$H(p) = \frac{1}{1 + \tau.p} \xrightarrow{Transformation} H(z) = \frac{1}{1 + \tau.\frac{1 - z^{-1}}{T_e}} = \frac{\frac{T_e}{\tau}.z}{(\frac{T_e}{\tau} + 1).z - 1}$$

Exemple : filtre passe-bas Matlab Euler

- Application : Filtre passe-bas 100Hz avec Fe=10kHz transformation de Euler

```
clear all;
close all;
clc;

fo=100; %100Hz
fe=10000; %10kHz pour la frÈquence d'Èchantillonnage
Te=1/fe;

%filtre analogique
num=[1];
den=[1 2*fo*pi];
[H,W] = freqs(num,den); %rÈponse frÈquentielle du filtre,
en Z
semilogx(W/(2*pi),abs(H)/max(abs(H))) %affichage en
frÈquence
hold

% filtre numÈrique euler
tau=2*fo*pi*j;
num_euler=[(Te*tau) 0];
denom_euler=[(Te*tau)+1 -1];
sys2=tf(num_euler, denom_euler, fe);
[H2,W2] = freqz(num_euler,denom_euler,20000); %rÈponse
frÈquentielle du filtre, en Z
semilogx(W2/(2*pi)*fe,(abs(H2)/max(abs(H2))),'m')
%affichage en frÈquence
legend('filre analogique','filre Euler');
```

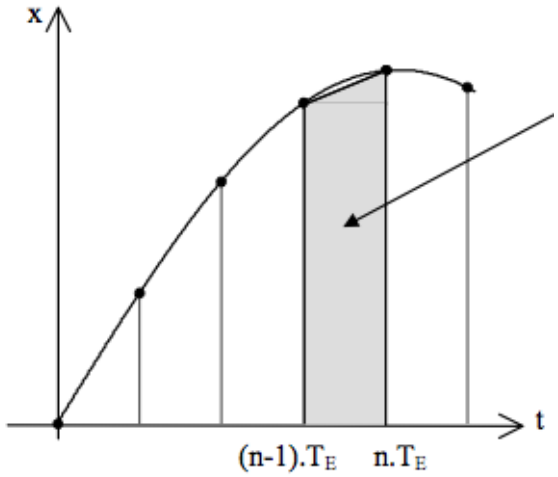


D. Transformation bilinéaire

Approximation de l'intégrale par la méthode des trapèzes

Enoncé du problème

- Approximation de la surface : intégration



$$\frac{T_e}{2}.\left(x(n)+x(n-1)\right)$$

$$x=\frac{dy}{dt}\longrightarrow y=\int x=y(n)-y(n-1)=\frac{T_e}{2}(x(n)+x(n-1))$$

- Transformation bilinéaire : $p \xrightarrow{Bilinéaire} \frac{2}{T_e}.\frac{1-z^{-1}}{1+z^{-1}}$

Méthode

1°) Détermination de la fonction de transfert continue du filtre dans le domaine continu, H(p)

2°) On applique la transformation bilinéaire pour trouver H(z)

$$p \xrightarrow{Bilinéaire} \frac{2}{T_e}.\frac{1-z^{-1}}{1+z^{-1}}$$

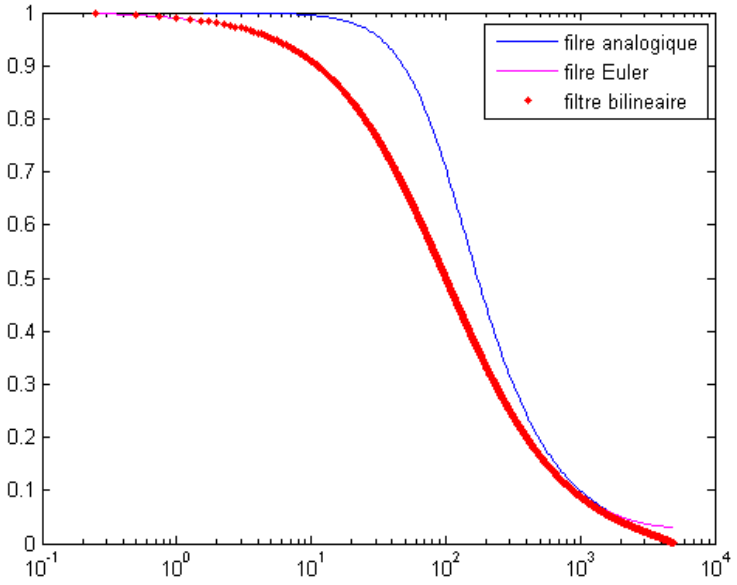
- Exemple : filtre passe-bas

$$H(p)=\frac{1}{1+\tau.p} \xrightarrow{Bilinéaire} H(z)=\frac{1}{1+\tau.\frac{2}{T_e}.\frac{1-z^{-1}}{1+z^{-1}}}=\frac{z+1}{z.(1+\tau.\frac{2}{T_e})+(1-\tau.\frac{2}{T_e})}$$

Exemple : filtre passe-bas Bilinéaire

- Application : Filtre passe-bas 100Hz avec Fe=10kHz transformation bilinéaire

```
clear all;
close all;
clc;

fo=100; %100Hz
fe=10000; %10kHz pour la frÈquence d'Èchantillonnage
Te=1/fe;

%filtre analogique
num=[1];
den=[1 2*fo*pi];
[H,W] = freqs(num,den); %rÈponse frÈquentielle du filtre, en Z
semilogx(W/(2*pi),abs(H)/max(abs(H))) %affichage en frÈquence
hold

% filtre numÈrique euler
tau=2*fo*pi*j;
num_euler=[(Te*tau) 0];
denom_euler=[(Te*tau)+1 -1];
sys2=tf(num_euler, denom_euler, fe);
[H2,W2] = freqz(num_euler,denom_euler,20000); %rÈponse frÈquentielle du filtre, en Z
semilogx(W2/(2*pi)*fe,(abs(H2)/max(abs(H2))),'m') %affichage en frÈquence
legend('filre analogique','filre Euler');

% filtre numÈrique bilineaire
tau=2*fo*pi*j;
num_bi=[1 1];
a=1+(2/(Te*tau));
b=1-(2/(Te*tau));
denom_bi=[a b];

sys3=tf(num_bi, denom_bi, fe);
[H3,W3] = freqz(num_bi,denom_bi,20000); %rÈponse frÈquentielle du filtre, en Z
semilogx(W3/(2*pi)*fe,(abs(H3)/max(abs(H2))),'.r') %affichage en frÈquence
legend('filre analogique','filre Euler','filtre bilineaire');
```



Synthèse de filtre numérique de type FIR

Enoncé du problème

- On cherche a réalisé un filtre non récursif dont la réponse impulsionnelle échantillonnée est h(n)

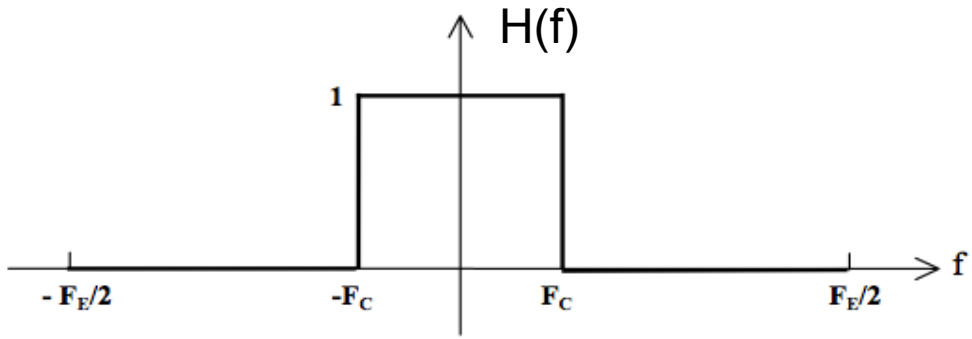
$$y(n) = \sum_{k=0}^{N-1} a_k . x(n-k)$$

$$H(z) = \sum_{k=0}^{N-1} a_k . z^{-k}$$

- 2 possibilités :
  A. On a accès à la réponse impulsionnelle du filtre continue
     - Echantillonnage temporel
  B. On a accès au gabarit fréquentiel du filtre continu
     - Echantillonnage fréquentiel

A. Echantillonnage de h(t)

- Gabarit souhaité de H(f)



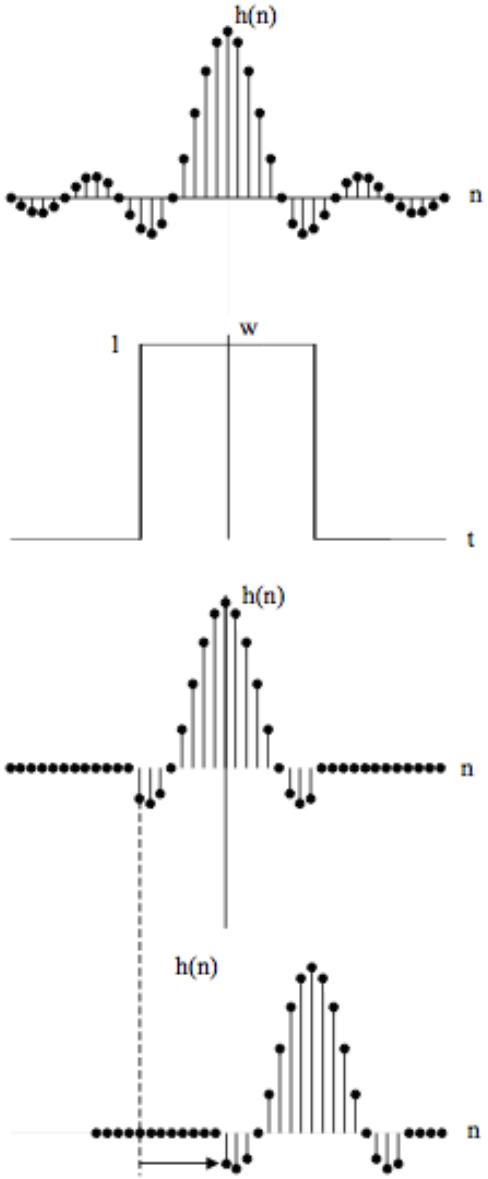
- Détermination de h(t) et h(n)

$$h(t) = TF^{-1}(H(f)) = \int_{-F_c}^{F_c} e^{j.2.\pi.f.t}.df = 2.F_c.\frac{\sin(2.\pi.F_c.t)}{2.\pi.F_c.t}$$

$$h(n) = 2.F_c.T_e.\frac{\sin(2.\pi.F_c.n.T_e)}{2.\pi.F_c.n.T_e}$$

A. Echantillonnage de h(t)

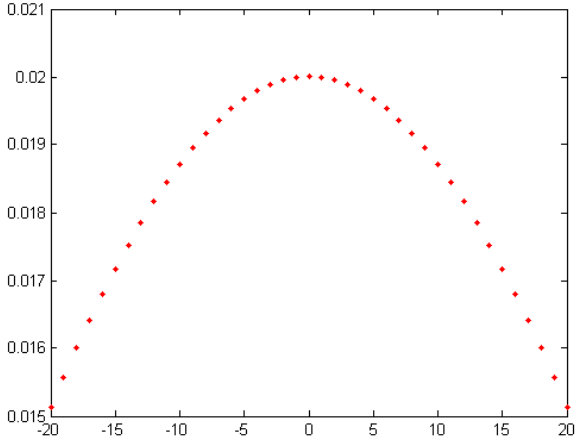
- h(n) est limité à un nombre fini d'échantillon
  - => effet de fenêtrage
  - On corrige par l'apodisation d'une fenêtre de correction (hamming, hann, etc.)
- h(n) est généralement symétrique h(-n)=h(n)
- h(n) est rendu causal par translation de la taille du filtre/2



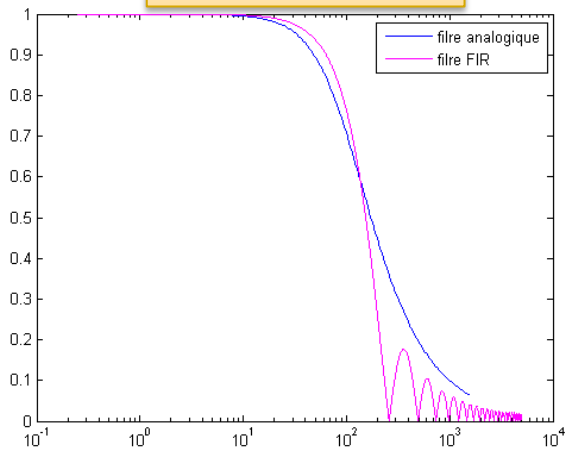
Exemple : filtre FIR passe-bas

- Filtre passe-bas 100Hz avec Fe=10kHz
  - =>Fc/Fe=0.01

```
clear all;
close all;
clc;

fo=100; %100Hz
fe=10000; %10kHz pour la frÈquence d'Èchantillonnage
Te=1/fe;

%filtre analogique
num=[1];
den=[1 2*fo*pi];
[H,W] = freqs(num,den); %rÈponse frÈquentielle du filtre, en Z
semilogx(W/(2*pi),abs(H)/max(abs(H))) %affichage en frÈquence
hold

% filtre numÈrique N coefficients
N=200;
n=-N:1:N;
h=2*fo*Te*sinc(2*n*fo*Te);

sys2=tf(h, 1, fe);
[H2,W2] = freqz(h,1,20000); %rÈponse frÈquentielle du filtre, en Z
semilogx(W2/(2*pi)*fe,(abs(H2)/max(abs(H2))),'m') %affichage en frÈquence
legend('filre analogique','filre FIR');

figure
plot(n,h,'.r');
```



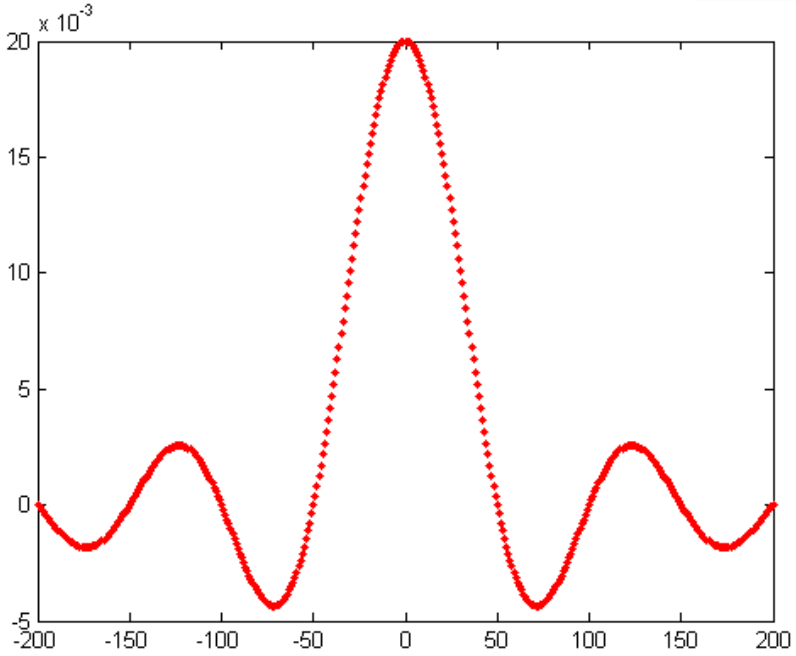
40 coefs

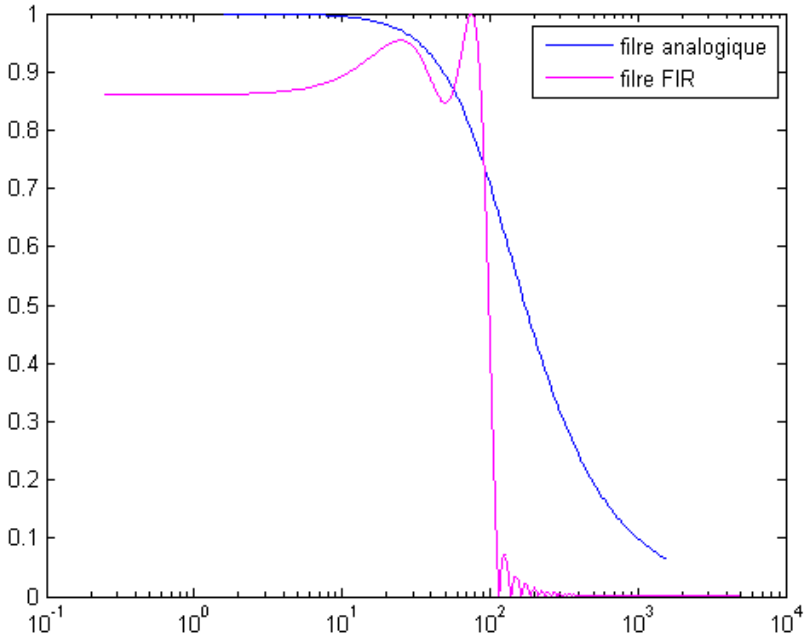


Exemple : filtre FIR passe-bas

- Filtre passe-bas 100Hz avec Fe=10kHz =>Fc/Fe=0.01

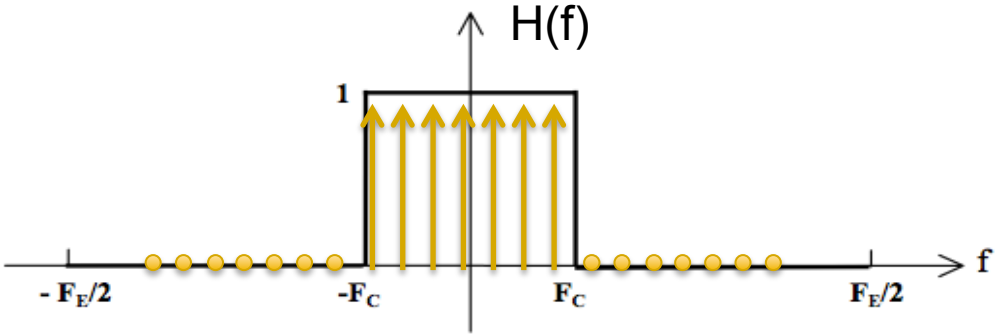
400 coefs





B. Echantillonnage fréquentielle H(f)

- Gabarit souhaité de H(f)



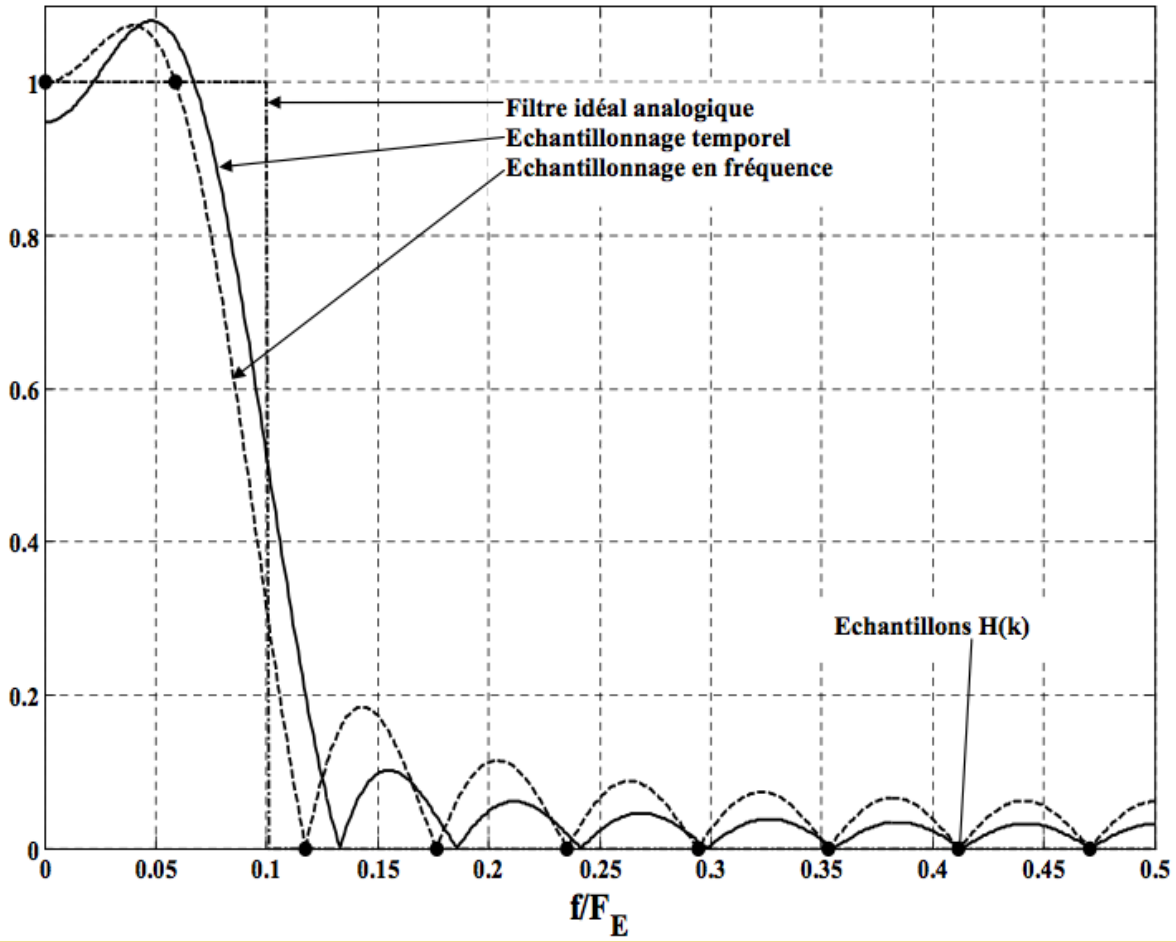
- Détermination de He(t) et h(n) par TFD^{-1}

$$h(n) = TFD^{-1}(H(\frac{k.f_e}{N}))$$

$$h(n) = \frac{1}{N}.\sum_{k=-\frac{N}{2}}^{\frac{N}{2}} H(k).e^{\frac{j.2.\pi.k}{N}}$$

B. Echantillonnage fréquentielle H(f)

- Exemple



Partie 5

Architecture des filtres RIF et RII

Architecture des FIR

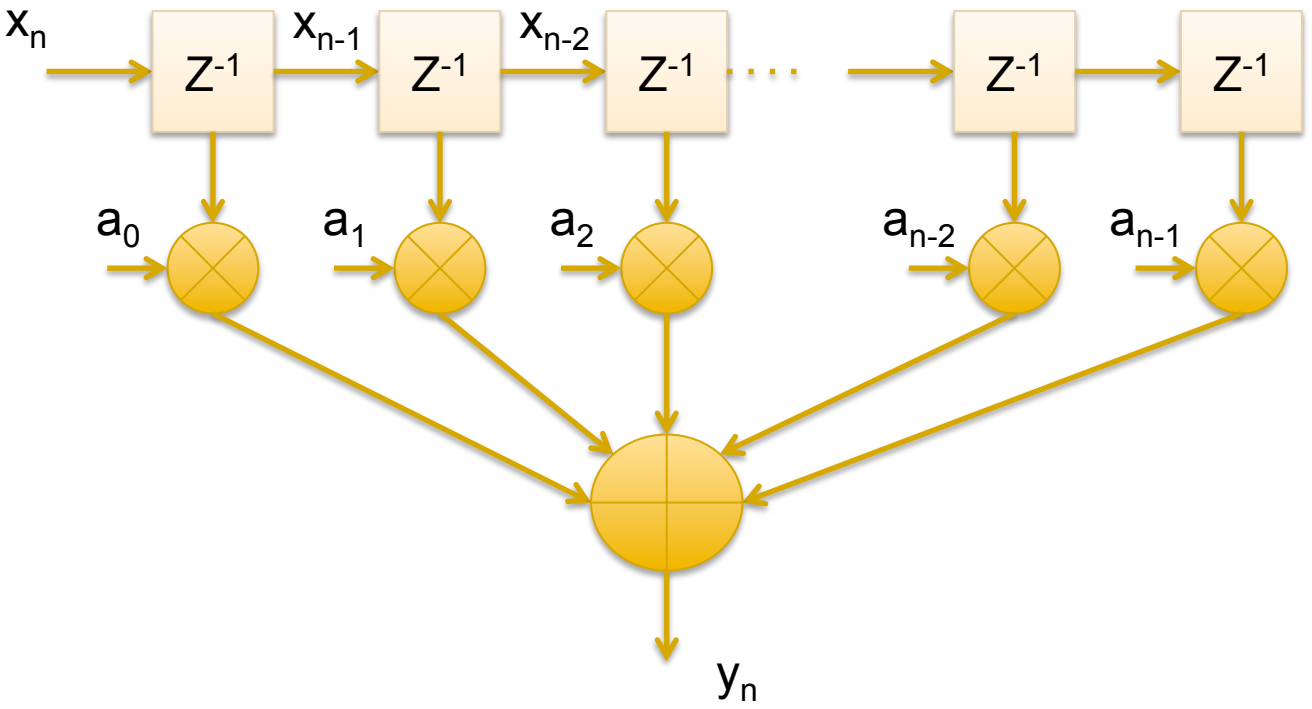
- Equation :

$$y(n) = \sum_{k=0}^{N-1} a_k . x(n-k)$$

x_n

$$H(z) = \sum_{k=0}^{N-1} a_k . z^{-k}$$

- Architecture //

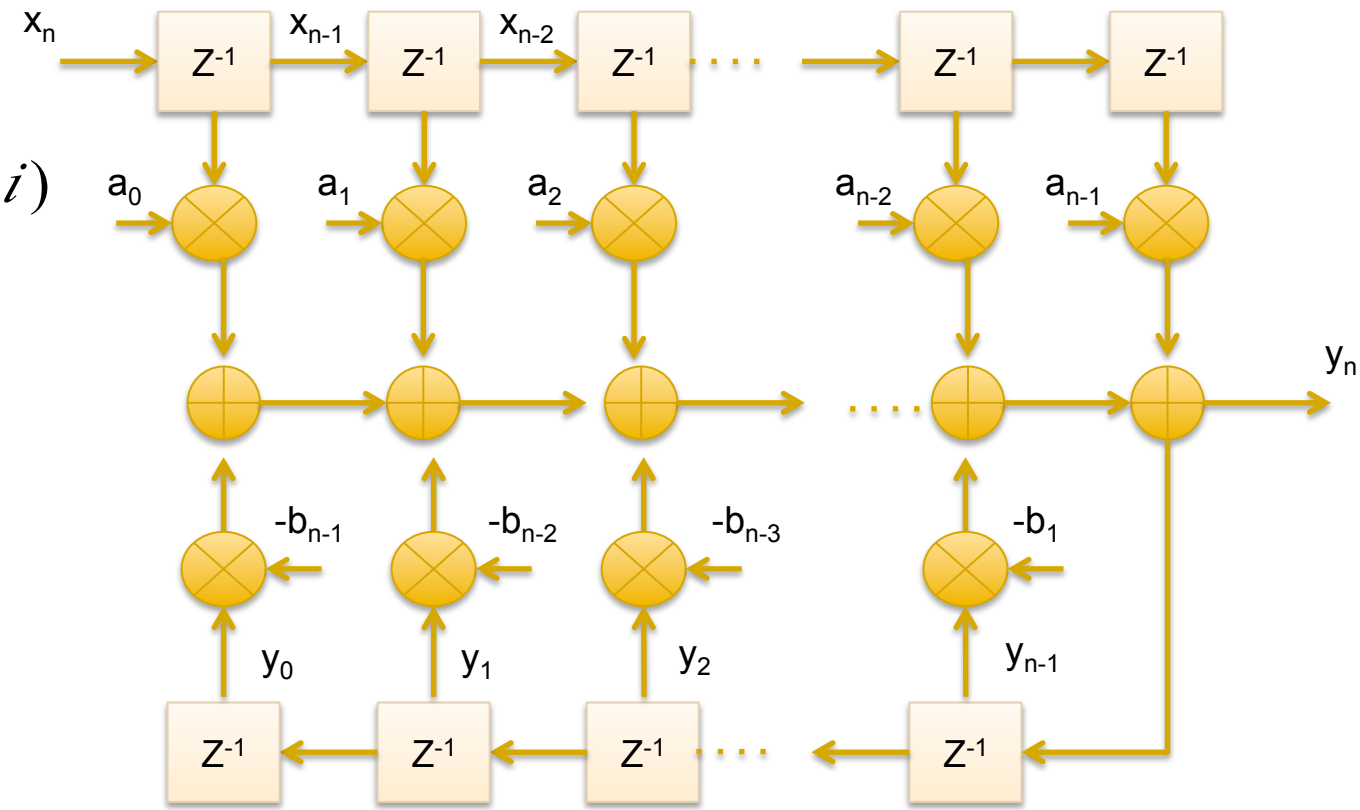


Architecture des RII

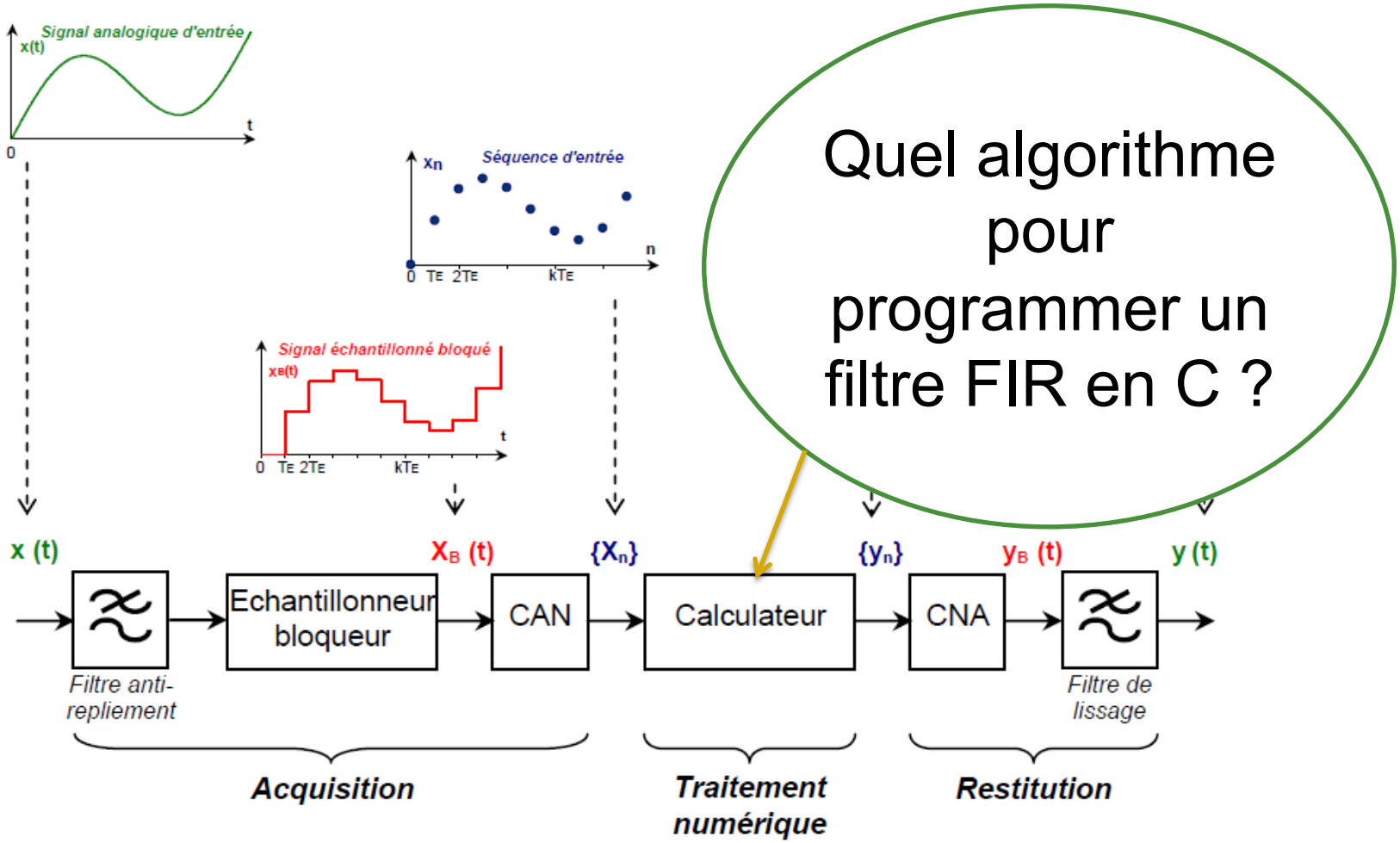
- Equation :

$$y(n) = \sum_{i=0}^{N-1} a_i x(n-i)$$

$$- \sum_{i=1}^{M-1} b_i y(n-i)$$



Algorithme filtre FIR



Algorithme filtre FIR

- Le filtrage est basé sur l'équation : $y(n) = \sum_{k=0}^{N-1} a_k . x(n-k)$
- L'opération de filtrage correspond à une opération de convolution temporelle
- Exemple : filtre h(n)={-1;1;1;-1}

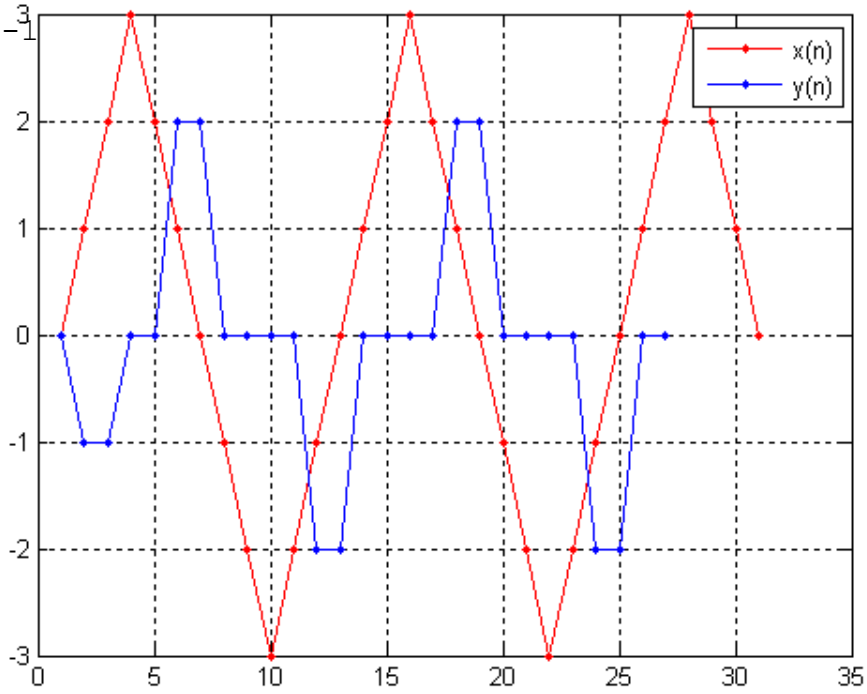
n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
X(n)	0	1	2	3	2	1	0	-1	-2	-3	-2	-1	0	1	2	3	2	1	0	-1
Y(n)	0	-1	-1	0	2	-2	0	0	0	0	-2	-2	0	0	0	0	2	2	0	0

Algorithme en C filtre FIR

```
….

x_buffer[n] = nouveau_sample;
for( n = (NB_COEF-1) ; n >= 0 ; n-- )
            y += coef[NB_COEF-1-n] * x_buffer[n];

for( n =0 ; n< NB_COEF-1 ; n++ )
             x_buffer[n] = x_buffer[n+1];

fprintf(stderr,"\n le data filtré est %f",y);

…
```

Algorithme filtre FIR : matlab

```
clear all;
close all;
clc;

%filtre analogique
x=[0 1 2 3 2 1 0 -1 -2 -3 -2 -1 0 1 2 3 2 1 0 -1
-2 -3 -2 -1 0 1 2 3 2 1 0];
h=[-1 1 1 -1];
plot(x,'.-r');
grid on
hold

%algo
n=length(x);
n_coef=length(h);

y(1)=x(1)*h(1);
y(2)=x(1)*h(2)+x(2)*h(1);
y(3)=x(1)*h(3)+x(2)*h(2)+x(3)*h(1);
y(4)=x(1)*h(4)+x(2)*h(3)+x(3)*h(2)+x(4)*h(1);

for p=n_coef+1:(n-n_coef)
    somme=0;
    for i=1:n_coef
        somme=somme+(h(i)*x(p-i));
    end
    y(p)=somme;
end
```




```
plot(y,'.-')
legend('x(n)','y(n)');
```

Partie 6

Outils de synthèse

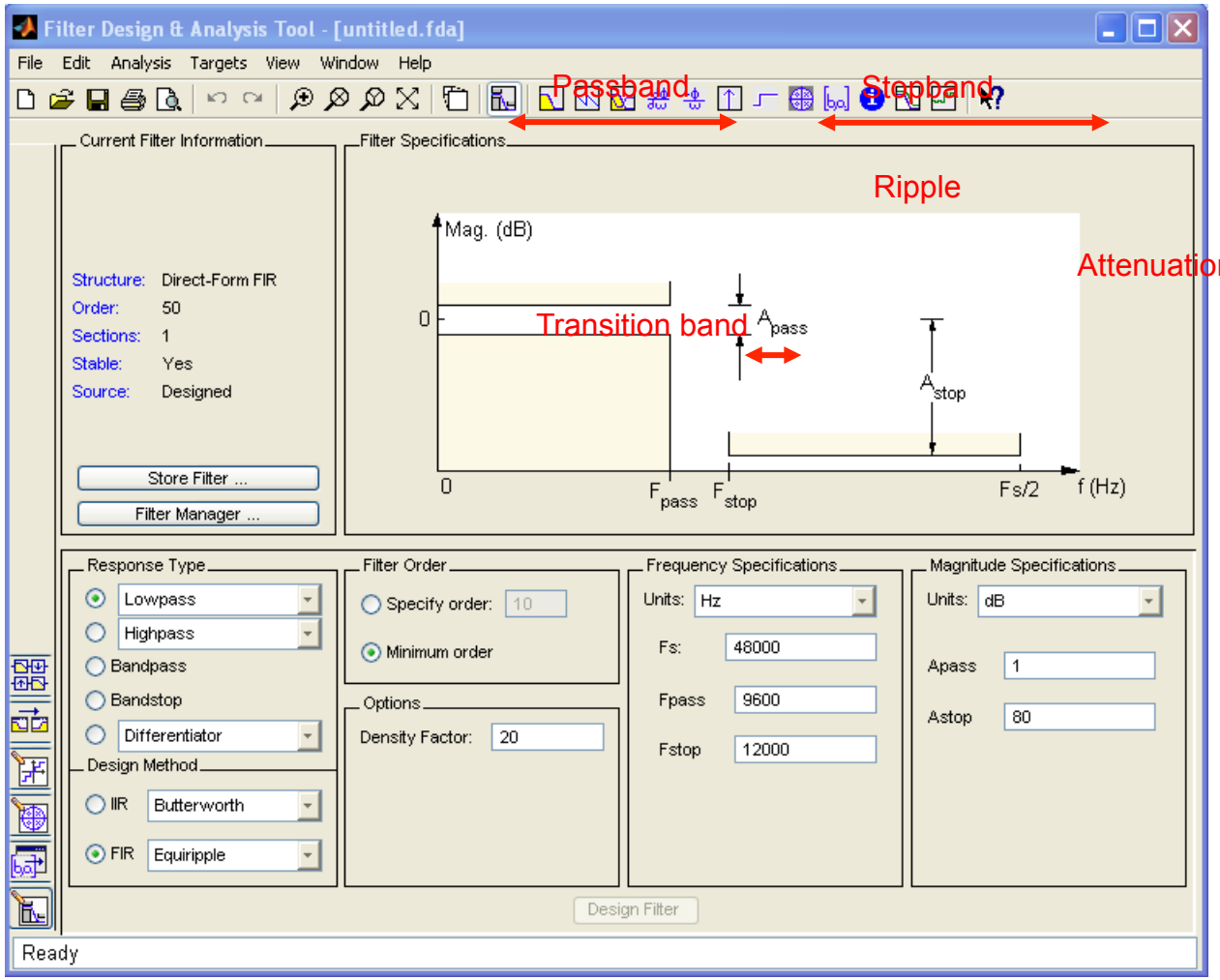
Outils de synthèse Matlab

- Signal Processing Tool : SPTOOL sous Matlab
  - Analyze signals
  - Design filters
  - Analyze (view) filters
  - Filter signals
  - Analyze signal spectra

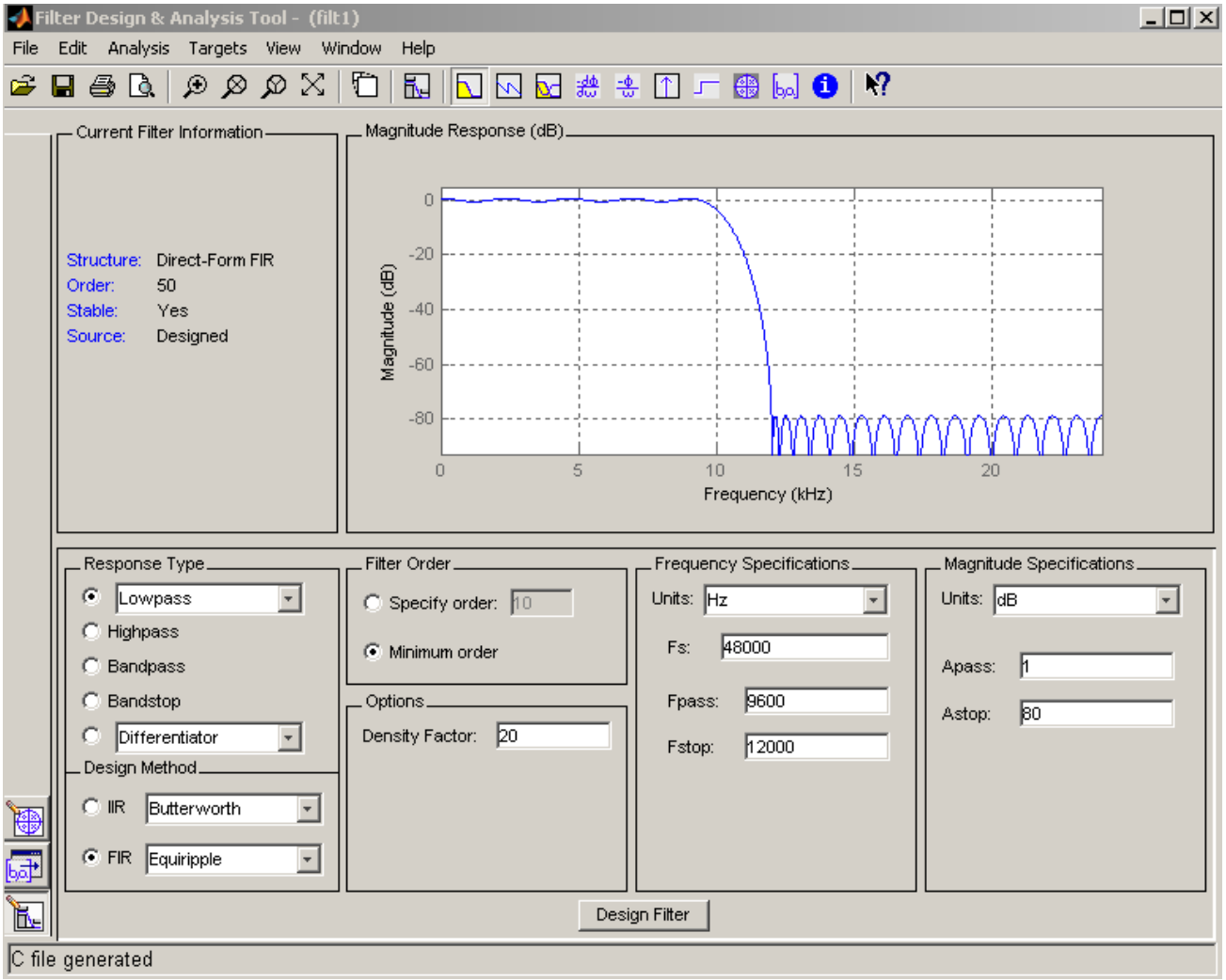
- FilterBuilder sous Matlab
  - Design filters
  - Génération automatique de VHDL

Response String	Description of Resulting Filter Design
`arbmag`	Arbitrary magnitude and phase filter
`bandpass` or `bp`	Bandpass filter
`bandstop` or `bs`	Bandstop filter
`cic`	CIC filter
`ciccomp`	CIC compensator
`diff`	Differentiator filter
`fracdelay`	Fractional delay filter
`halfband` or `hb`	Halfband filter
`highpass` or `hp`	Highpass filter
`hilb`	Hilbert filter
`isinclp`	Inverse sinc lowpass filter
`lowpass` or `lp`	Lowpass filter (default)
`notch`	Notch filter
`nyquist`	Nyquist filter
`octave`	Octave filter
`parameq`	Parametric equalizer filter
`peak`	Peak filter
`pulseshaping`	Pulse-shaping filter

Signal Processing Tool



Signal Processing Tool

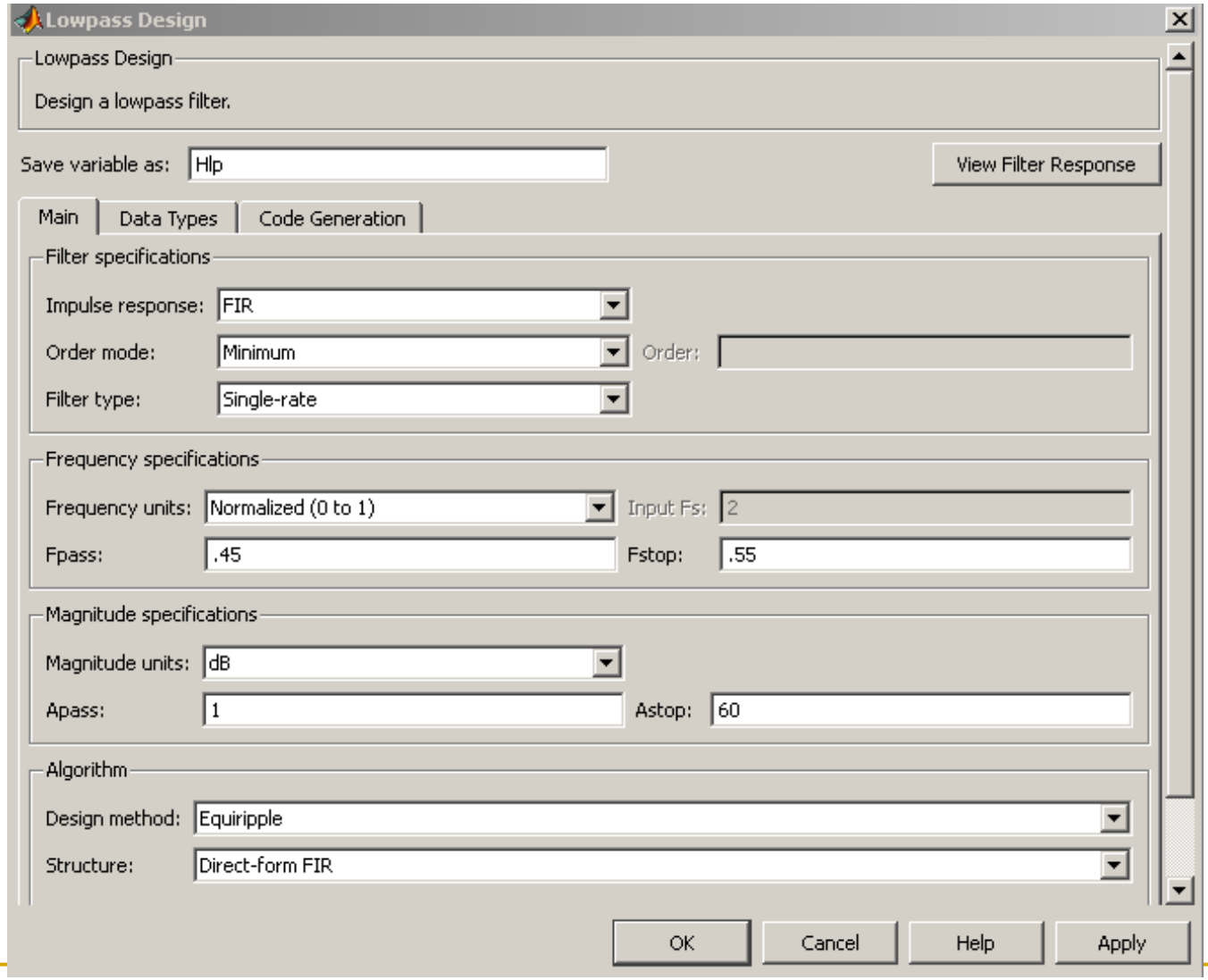


Signal Processing Tool

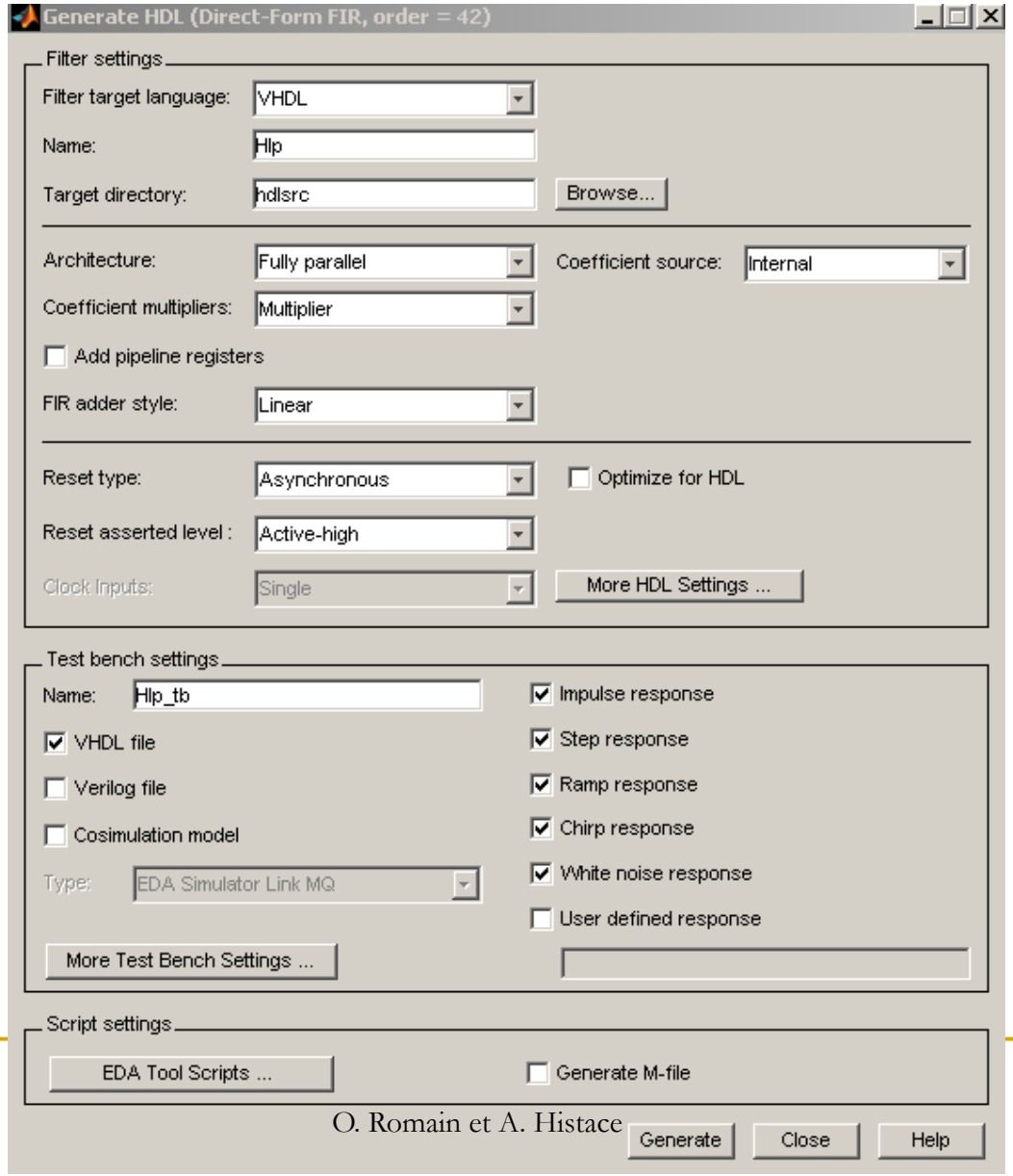
- Génération de fichier header de coefficients

```
 * Filter Coefficients (C Source) generated by the Filter Design and Analysis Tool
 *
 * Generated by MATLAB(R) 7.8 and the Signal Processing Toolbox 6.11.
 *
 * Generated on: 22-Jan-2013 16:14:40
 *
 */

/*
 * Discrete-Time FIR Filter (real)
 * -------------------------------
 * Filter Structure  : Direct-Form FIR
 * Filter Length     : 51
 * Stable            : Yes
 * Linear Phase      : Yes (Type 1)
 */

/* General type conversion for MATLAB generated C-code  */
#include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2009a\extern\include\tmwtypes.h
 */
const int BL = 51;
const real64_T B[51] = {
  -0.0009190982084683,-0.002717696026596,-0.002486952759832, 0.003661438383507,
    0.01365092523066,  0.01735116590109, 0.007665306190422,-0.006554718869642,
  -0.007696784037065, 0.006105459421394,  0.01387391574864,0.0003508617282909,
   -0.01690892543669,-0.008905642749159,  0.01744112950085,  0.02074504452761,
   -0.01229649425194, -0.03424086590958,-0.001034529605572,   0.0477903055208,
    0.02736303791485, -0.05937951883105, -0.08230702592923,  0.06718690943287,
     0.3100151770903,   0.4300478803435,   0.3100151770903,  0.06718690943287,
   -0.08230702592923, -0.05937951883105,  0.02736303791485,   0.0477903055208,
  -0.001034529605572, -0.03424086590958, -0.01229649425194,  0.02074504452761,
    0.01744112950085,-0.008905642749159, -0.01690892543669,0.0003508617282909,
    0.01387391574864, 0.006105459421394,-0.007696784037065,-0.006554718869642,
   0.007665306190422,  0.01735116590109,  0.01365092523066, 0.003661438383507,
  -0.002486952759832,-0.002717696026596,-0.0009190982084683
};
```

FilterBuilder

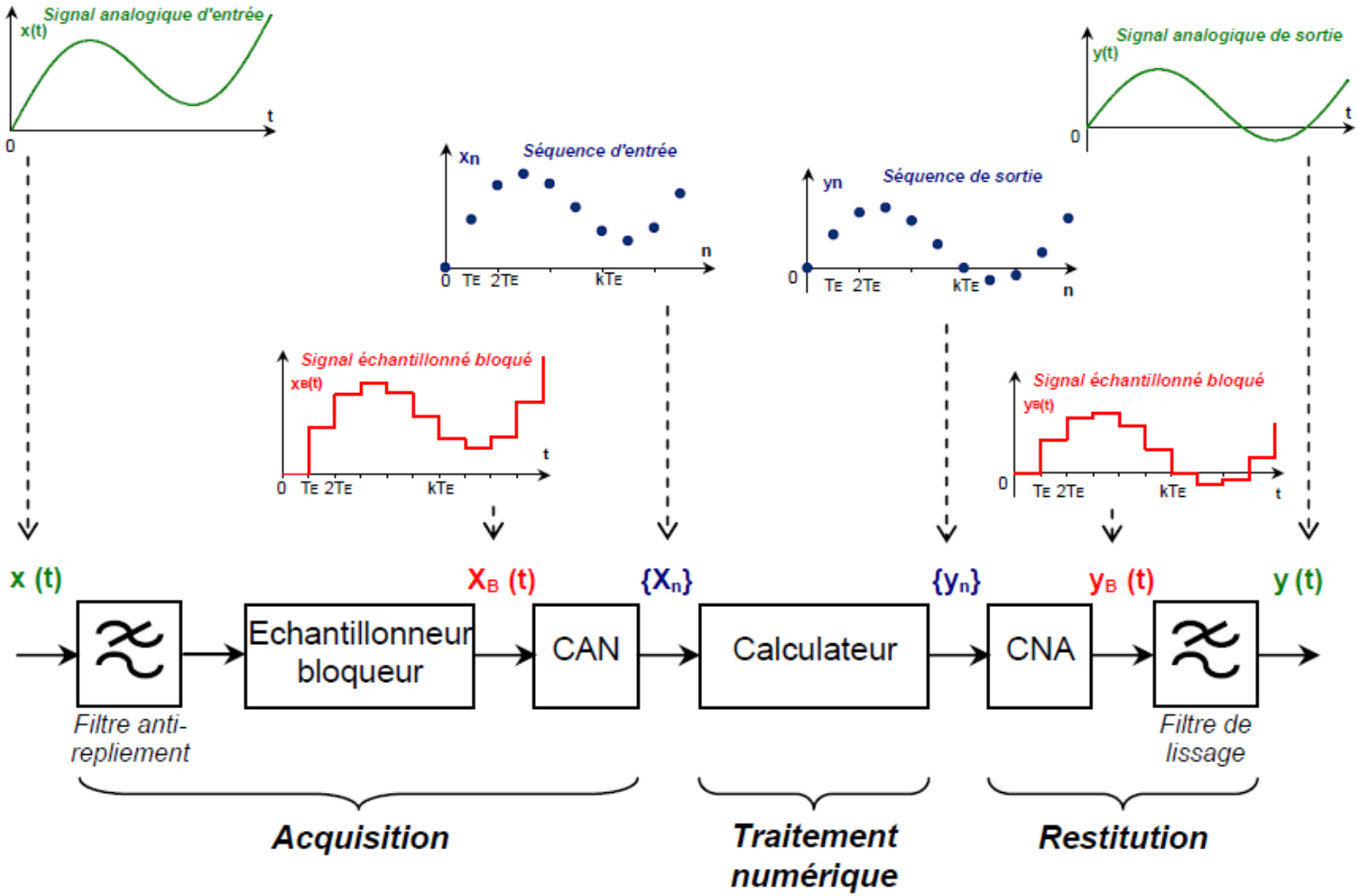


FilterBuilder : HDL generation



Conclusion

Structure TNS



Structure TNS