Premier Cycle – INSA de LYON

Ouverture thématique – 2ème année

Bases du Signal Signaux Numériques et Applications

Equipe enseignante :

Nom	Prénom	Dept	Laboratoire	E-mail
BENOIT-CATTIN	Hughes	TC	Creatis	Hugues.Benoit-Cattin@creatis.insa-lyon.fr
BRES	Stéphane	IF	LIRIS	stephane.bres@insa-lyon.fr
DELACHARTRE	Philippe	GE	Creatis	philippe.delachartre@creatis.insa-lyon.fr
FRIBOULET	Denis	PC	Creatis	denis.friboulet@creatis.insa-lyon.fr
LELEVE	Arnaud	GI	ICCT	Arnaud.Leleve@insa-lyon.fr
VRAY	Didier	PC	Creatis	didier.vray@creatis.insa-lyon.fr

Bibliographie :

- Max, Jacques, Lacoume, Jean-Louis. Méthodes et techniques de traitement du signal. Paris : Dunod, 2000, 355 p. ISBN 2-10-005332-9.

- Lacoume, Jean-Louis, Max, Jacques. Méthodes et techniques de traitement du signal 2ème cycle, école d'ingénieurs. 5ème édition. Paris : Dunod, 2004, 355 p. ISBN 2-10-048331-5.

- F. De Coulon, Théorie et traitement des signaux, Dunod, 1984.

- Oppenheim, Alan V., Willsky, Alan S., Signals & systems, 2nd edition. - New Jersey : Prentice-Hall , 1997 . - XXX-957 p – ISBN 0-13-651175-9.

- B. Mulgrew et al., "Digital Signal Processing: Concepts and Applications", Ed: Mac Millan Press, 1999, 355p.

Département PREMIER CYCLE

8 allée Lumière – 69621 Villeurbanne Cedex - France

Tel : 33(0)4 72 43 83 84 Télécopie : 33(0)4 72 43 64 34

E_mail : Site web : http://pc.insa-lyon.fr

Identification
Code : **2PC_OT439**
ECTS : 2
☐Annuel ☒Semestriel ☒1er semestre ☐2è semestre
☐Obligatoire ☒Optionnel
Horaires
Cours : 12h
TD: 4h
TP: 14h
Projet :
Travail personnel : 10h
Evaluations
Contrôle continu :
Devoir Synthèse :
Soutenance :
Supports pédagogiques
Polycopiés Cours, TD et TP OUI
Fichier PPT en ligne OUI
Logiciel : Matlab
Enseignant Responsable
Vray Didier Friboulet Denis
Département de rattachement *PC*
Tel 04 72 43 87 84 04 72 43 89 75

2ème ANNEE

Intitulé du Module : Signaux numériques et applications

Module proposé par une équipe d'enseignants issus des départements GE, GI, IF, PC, TC

Nombre d'élèves max. : 24

- **Objectifs et compétences :**

A partir d'exemples concrets tirés des télécommunications, du traitement de la parole, de l'imagerie médicale, des techniques physiques et d'instrumentations, ce module d'Ouverture Thématique fournira aux étudiants les bases du traitement du signal qu'ils pourront mettre à profit quelle que soit la spécialité qu'ils décideront d'aborder ensuite pour leur formation d'ingénieur. Le module se déroule successivement dans 4 départements de l'INSA (GE, GI, IF, TC).

- **Programme :**

Cours et TD : Signaux, exemples, analyse fréquentielle, Transformée de Fourier, analyse spectrale, filtrage, échantillonnage, filtrage numérique, compression.

Séances Machines : logiciel Matlab, signaux numériques, représentation de signaux en temps et en fréquence, transformée de Fourier Discrète, analyse d'un signal inconnu, filtrage, fenêtres de pondération (hamming…), représentation temps-fréquence, modulation d'amplitude, modulation de fréquence, analyse signal de parole, logiciel SPTOOLS, applications en Telecom, logiciel SIMULINK.

- **Contrôle :** évaluation des projets pratiques et contrôle de connaissances 2h

- **Travail Personnel :** 10h pour l'ensemble du module

- **Pré-requis :** Aucun pour un étudiant du Premier Cycle de l'INSA

- **Bibliographie :**

- Lacoume, Jean-Louis, Max, Jacques. Méthodes et techniques de traitement du signal 2ème cycle, école d'ingénieurs, Paris, Dunod, 2004, 355 p.
- F. De Coulon, Théorie et traitement des signaux, Dunod, 1984.
- Oppenheim, Alan V., Willsky, Alan S., Signals & systems, 2nd edition. - New Jersey : Prentice-Hall, 1997, 957 p
- B. Mulgrew et al., "Digital Signal Processing: Concepts and Applications", Ed: Mac Millan Press, 1999, 355p.

Mail didier.vray@insa-lyon.fr, denis.friboulet@insa-lyon.fr

Date d'actualisation de la fiche OT : Juin 2010

Institut National des Sciences Appliquées de Lyon

1

Traitement du signal et applications

Introduction

Denis Friboulet – Didier Vray

2

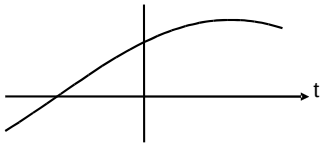
SIGNAL ?

- **Domaines d'application :**
  - communication, aéronautique, astronautique, acoustique, contrôle de processus chimique, ingénierie biomédicale, traitement de la parole, économie, météorologie...
  - information représentée sous forme de signal

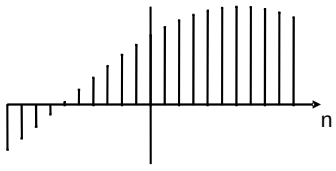
- **Signal :**
  - fonction de une ou plusieurs variables indépendantes : f(x,y,z)
  - représente / contient information sur un phénomène (physique)

- **Types de signaux**

Signaux continus: f(t), t ∈ **R**



Signaux discrets: f[n], n ∈ **Z**

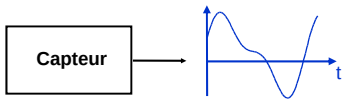


- **Dimension du signal :**
  - 1D : f(t) f[n]
  - 2D :f(u,v) f[i,j]
  - 3D : f(u,v,w) f[i,j,k]
  - etc...

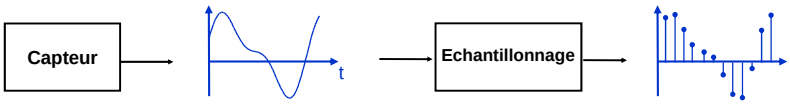
SIGNAL ?

3

- **Mesures**
  - Capteur, statistiques....
  - Développement des capteurs : microphone, céramiques piézoélectriques, antennes, capteurs CCD, de pression



- **Acquisition numérique :**

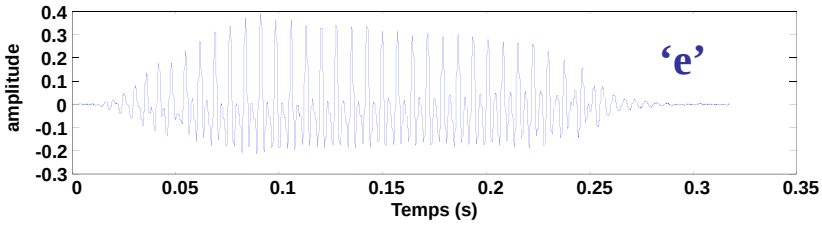


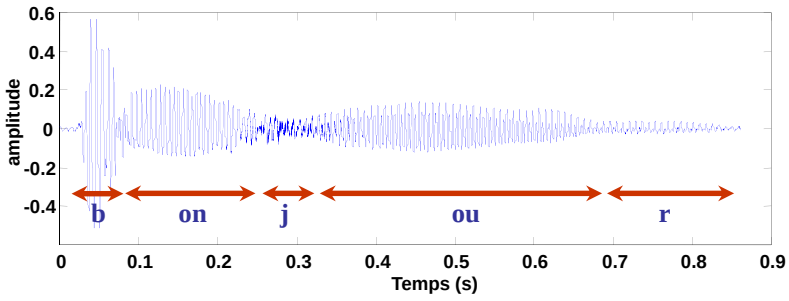
  - Stockage
  - Traitement numérique par ordinateur
  - Perte d'information ?
  - Importance des techniques numériques

EXEMPLES DE SIGNAUX

4

Signal de parole

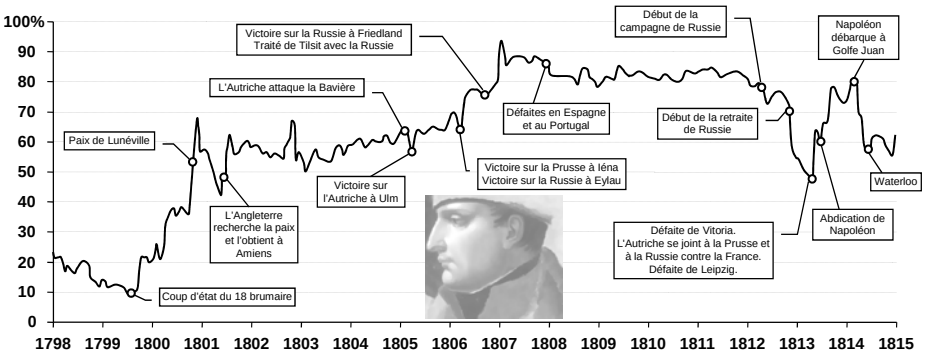




EXEMPLES DE SIGNAUX

5

Évolution de la rente 5% sous le consulat et le 1er empire

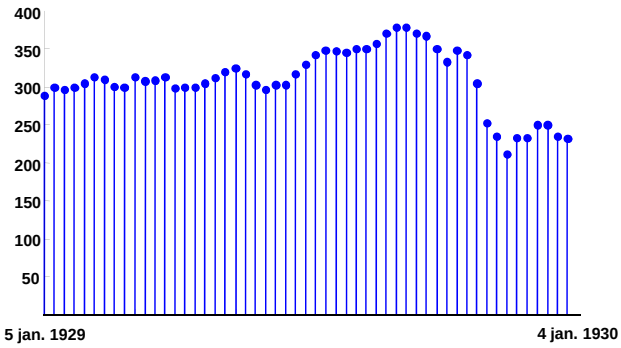


© Le Monde 28/01/1997

EXEMPLES DE SIGNAUX

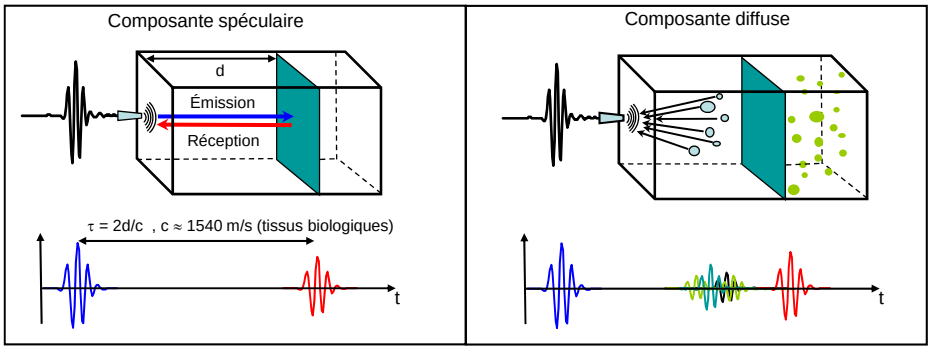
6

Évolution hebdomadaire du Dow Jones du 5 janvier 1929 au 4 janvier 1930

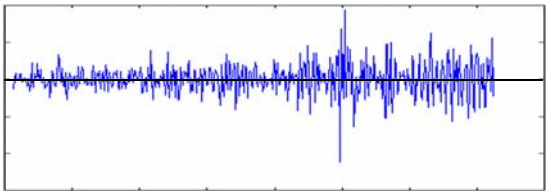


EXEMPLES DE SIGNAUX

Signaux ultrasonores

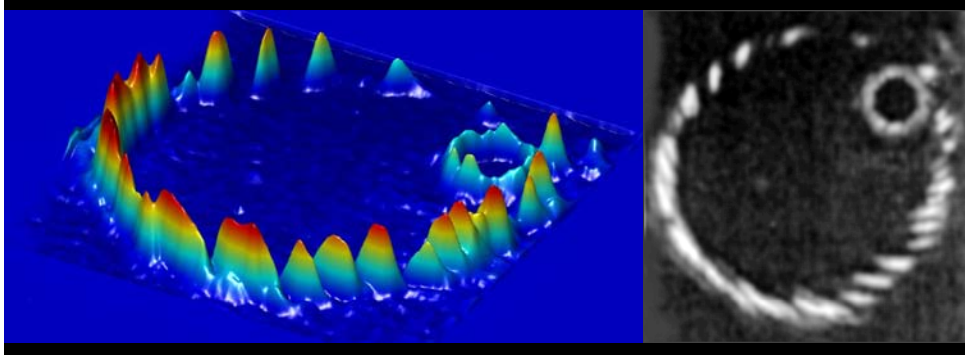


Signal réel



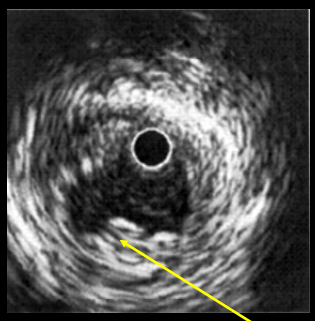
EXEMPLES DE SIGNAUX

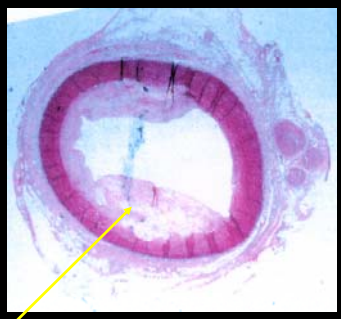
Images ultrasonores: échographie intravasculaire



EXEMPLES DE SIGNAUX

Images ultrasonores: échographie intravasculaire

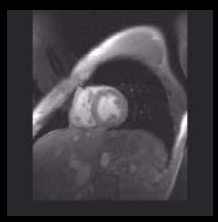




Plaque artérielle

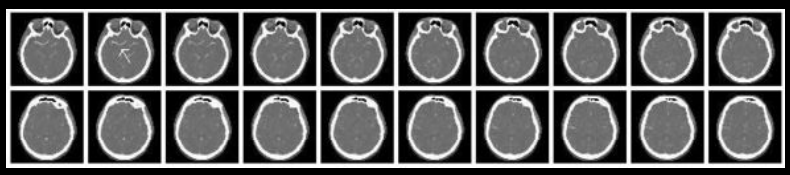
EXEMPLES DE SIGNAUX

Images cardiaques échographie ultrasonore et en IRM



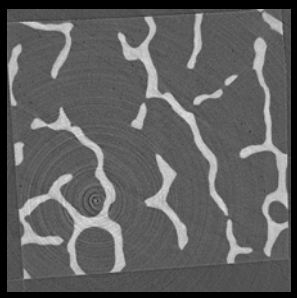
EXEMPLES DE SIGNAUX

Tomographie X



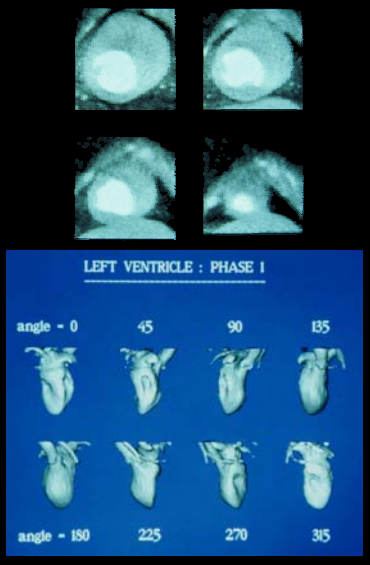
EXEMPLES DE SIGNAUX

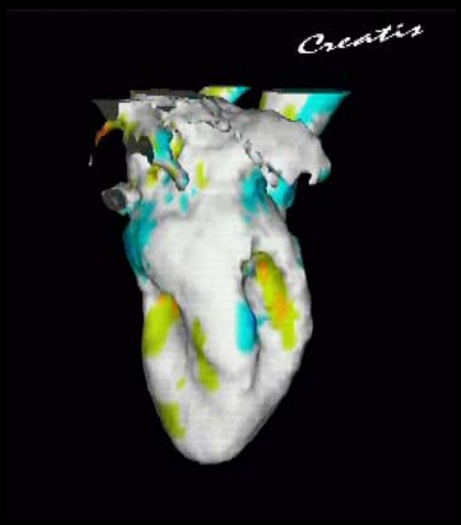
Tomographie par rayonnement X synchrotron



EXEMPLES DE SIGNAUX

Tomographie X



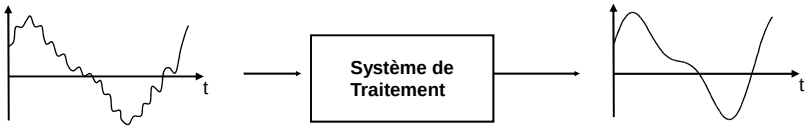


TRAITEMENT DU SIGNAL ?

14

- **Buts :**
  - Modélisation : représentation d'un phénomène (caractérisation, prédiction...)
  - Analyse : extraction d'information (mesure, compression, détection, reconnaissance...)
  - Filtrage, restauration : transformation du signal (minimisation du bruit, suppression de parasite...)
  - Etc.

- **Notion de système de traitement :**

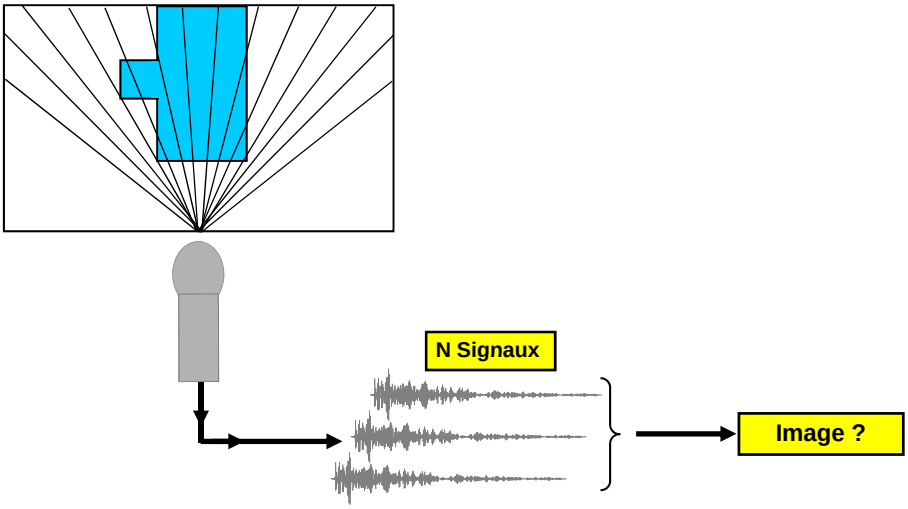


- **Notion de domaine de représentation :**
  - Transformée de Fourier
  - Modèles autorégressifs, transformée de Laplace, transformée de Wigner, transformée en ondelettes, transformée cosinus....

EXEMPLES DE TRAITEMENT

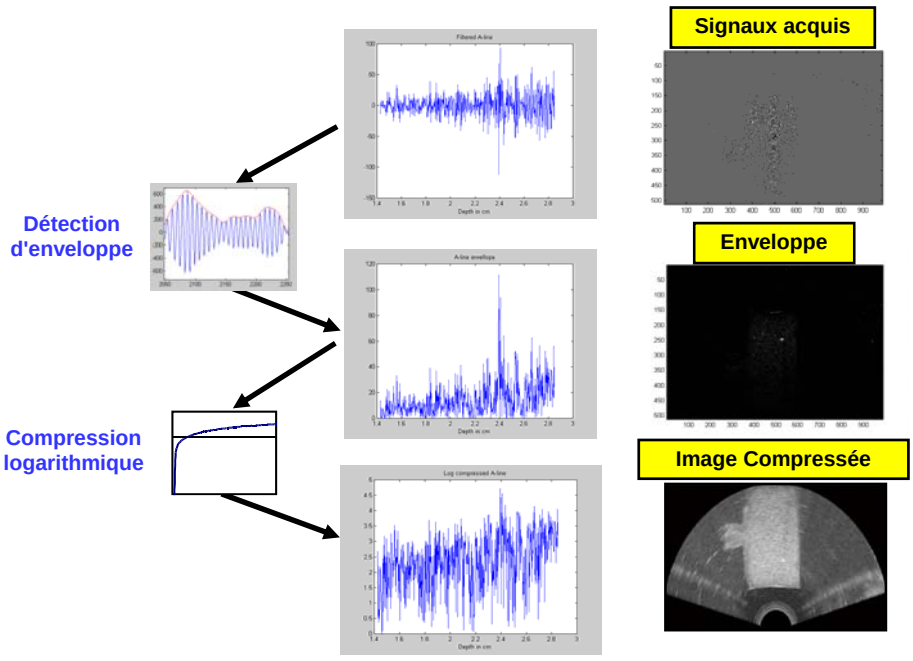
15

Imagerie ultrasonore

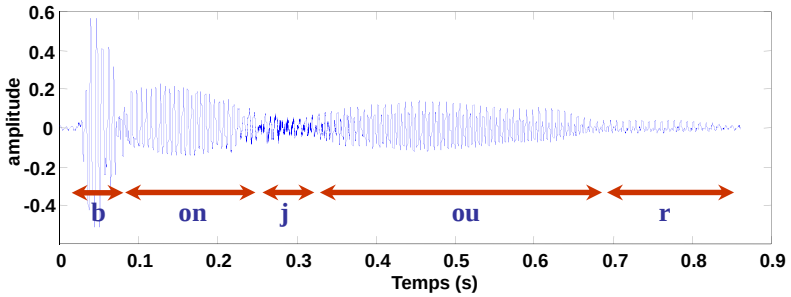


EXEMPLES DE TRAITEMENT

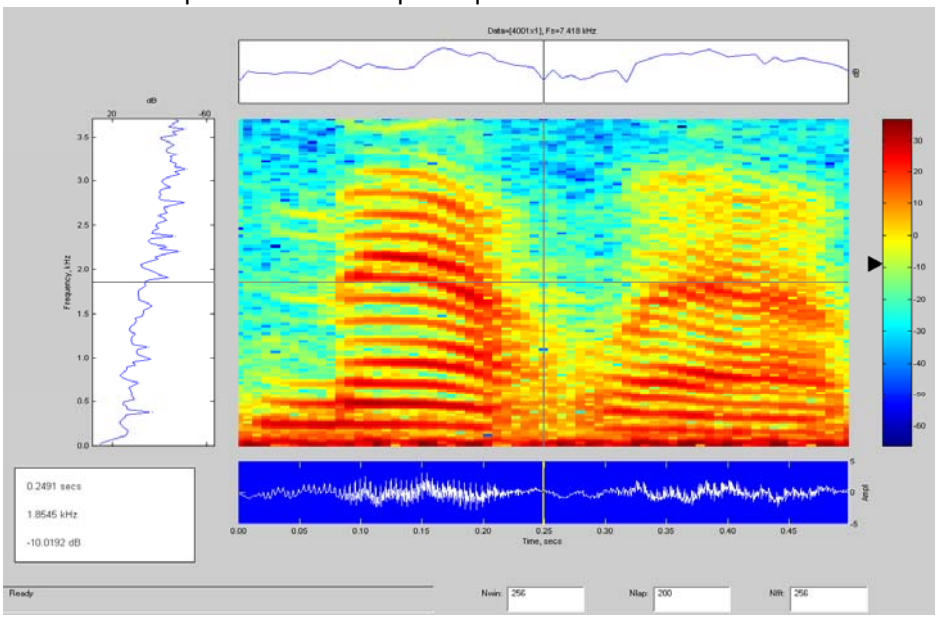
16



Signal de parole : analyse ?

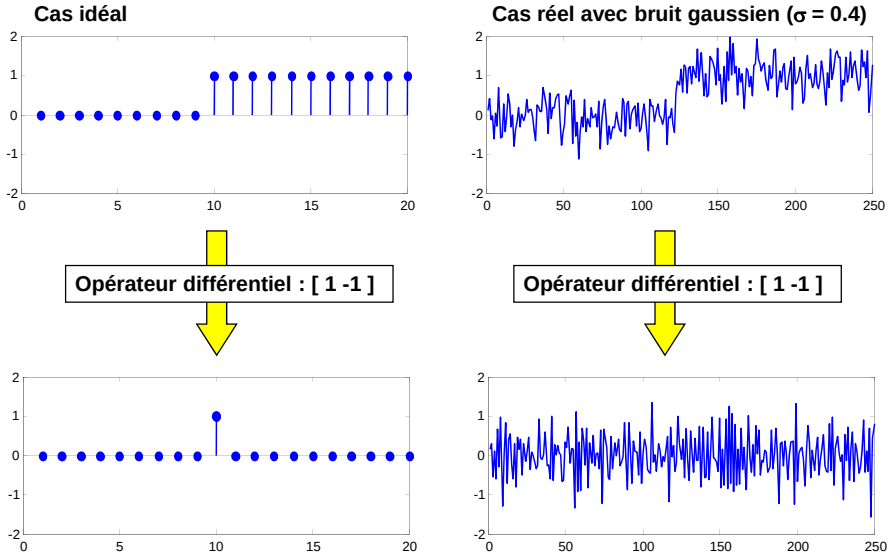


Domaine de représentation : temps-fréquence



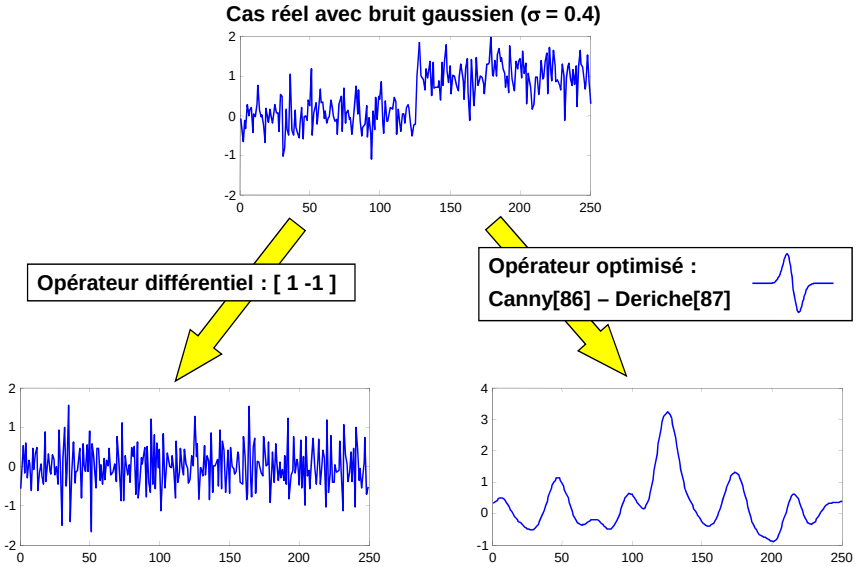
EXEMPLES DE TRAITEMENT

Détection de rupture



EXEMPLES DE TRAITEMENT

Détection de rupture



Détection de rupture

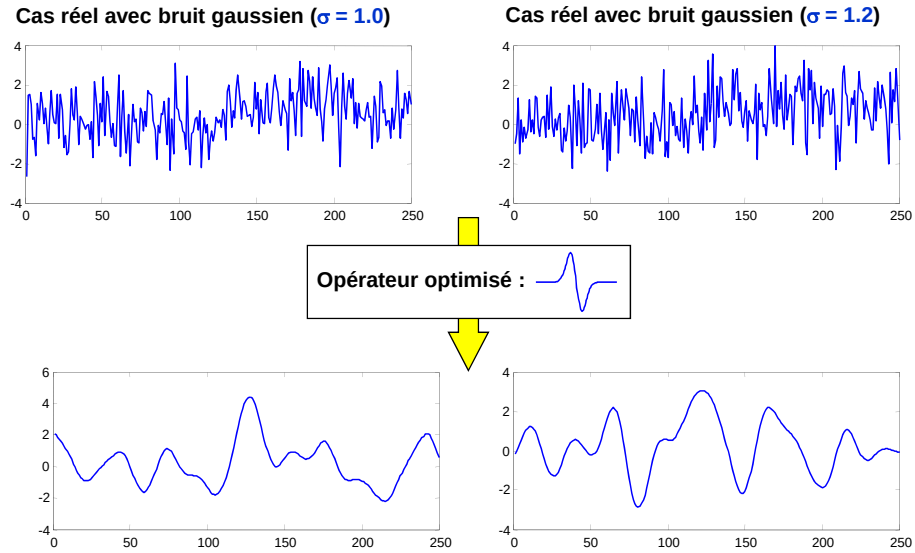
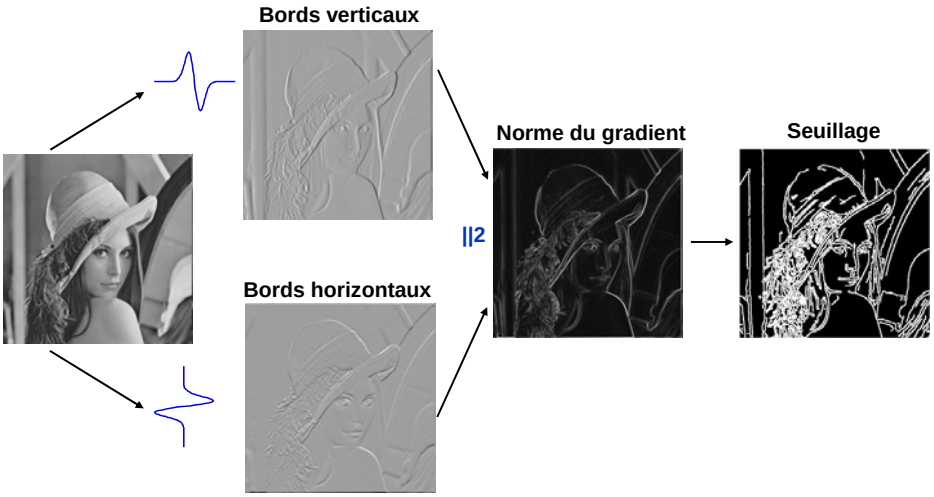


Image : Détection de contours

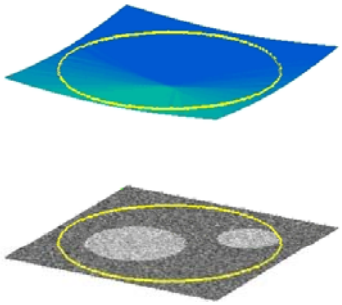


EXEMPLES DE TRAITEMENT

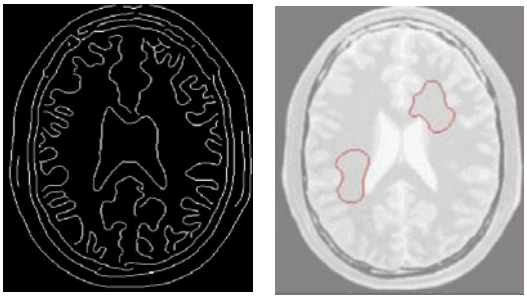
23

Détection de contour par ensemble de niveau basé sur les dérivées (gradient)

Ensemble de niveau: Principe d'évolution



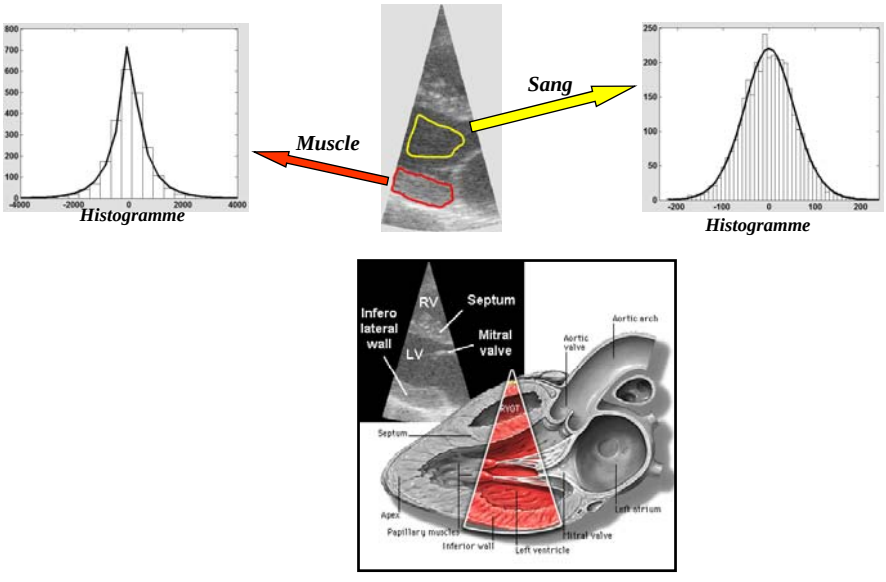
Evolution basée sur le gradient d'une image IRM



EXEMPLES DE TRAITEMENT

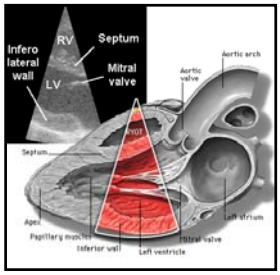
24

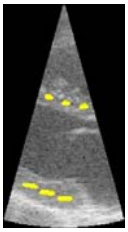
Détection de contour par ensemble de niveau basé sur les statistiques

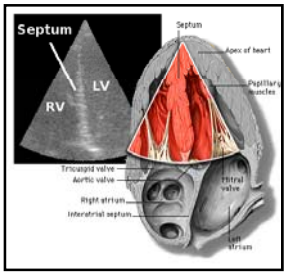


EXEMPLES DE TRAITEMENT

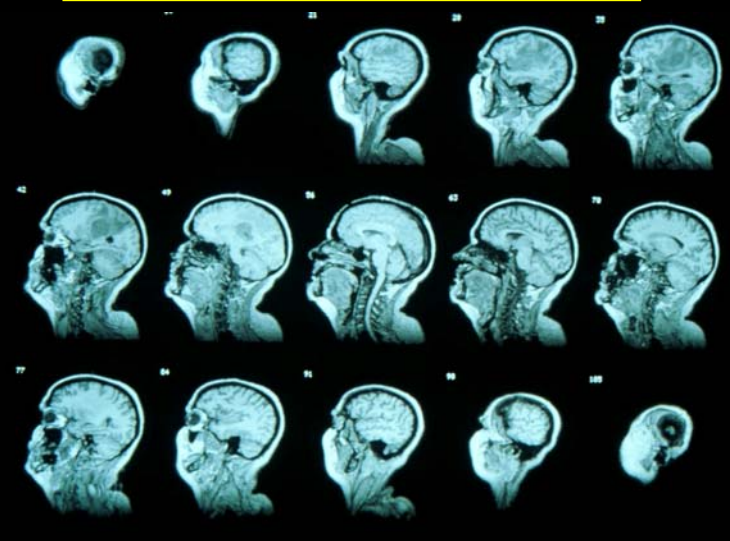
Détection de contour par ensemble de niveau basé sur les statistiques







EXEMPLES DE TRAITEMENT

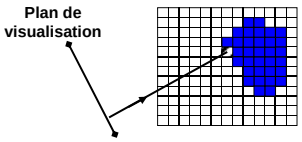


→ Visualisation de l'information ?

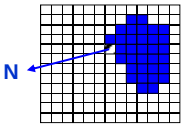
EXEMPLES DE TRAITEMENT

➢ **Visualisation : Rendu de volume par lancer de rayon**

1. Intersection avec l'objet : Tracé incrémental d'un rayon dans le volume discret

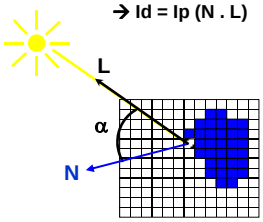


2. Estimation de la normale à l'objet : opérateur différentiel discret dans les 3 directions

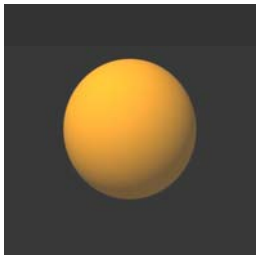


3. Calcul de la lumière émise au point d'intersection : modèle de diffusion

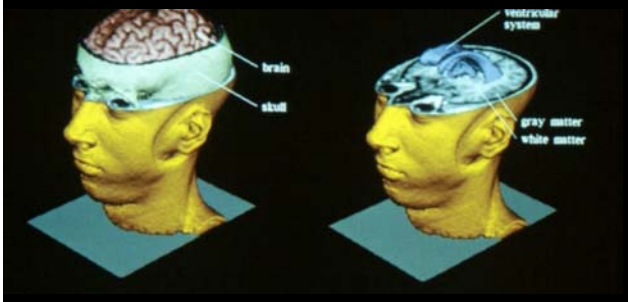
→ Id = Ip (N . L)

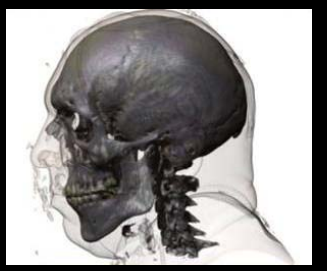


4. Application des étapes 1, 2 et 3 à l'ensemble des rayons partant du plan de visualisation

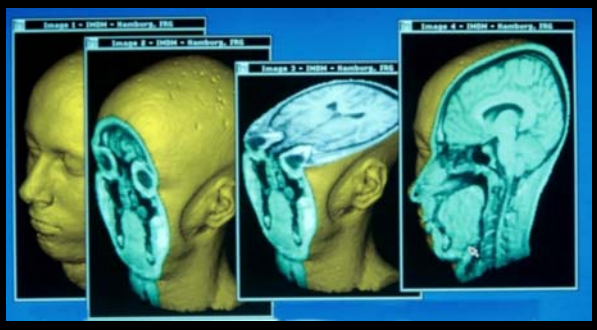


EXEMPLES DE TRAITEMENT





© Joe Kniss, Utah



EXEMPLES DE TRAITEMENT

BIBLIOGRAPHIE

Ouvrages généraux

- Bernard MULGREW, "Digital Signal Processing, concepts and applications", Palgrave MacMillan, 2003
- Alan OPPENHEIM, A. SCHAFER, "Discrete-time Signal Processing", PRENTICE HALL, 1999
- Alan OPPENHEIM, A. WILLSKY, "Signals and Systems", PRENTICE HALL, 1997
- J. Mc CLELLAN, "DSP First, A multimedia approach", PRENTICE HALL, 1999
- François de COULON, "Théorie et traitement des signaux", DUNOD, 1984
- Murat KUNT, "Traitement Numérique des signaux", 1981

Applications ou pour aller plus loin

- J. MARS, J.-L. LACOUME, "Traitement du signal pour géologues et géophysiciens", TECHNIP 2004, Tome 3 : Techniques avancées et Tome 2 : Techniques de base, Tome 1 : Prospection sismique
- M. BELLANGER, "Traitement numérique du signal : Théorie et pratique", 8ème édition, DUNOD, 2006
- S. MARPLE, "Digital spectral analysis", PRENTICE-HALL, 1987

31

Signaux de base

Opération élémentaires sur les signaux

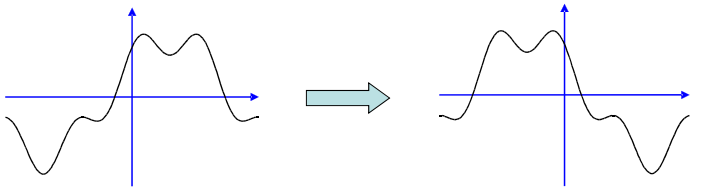
32

TRANSFORMATIONS DE LA VARIABLE INDEPENDANTE

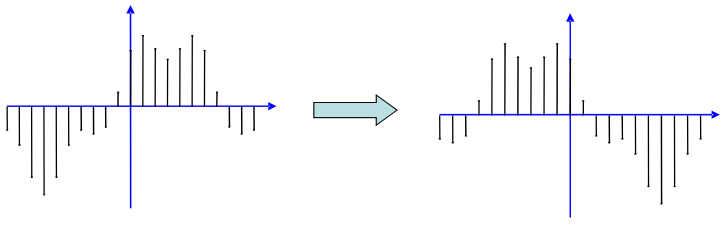
- **Opérations de base sur les signaux:**
  - Retournement, décalage, changement d'échelle, etc.

  → transformation de la variable indépendante

- **Retournement**



f(t) **f(-t)**

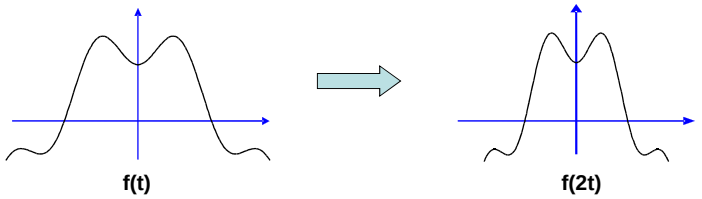


f[n] **f[-n]**

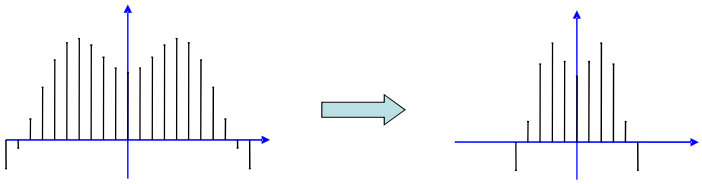
TRANSFORMATIONS DE LA VARIABLE INDEPENDANTE

33

- **Changement d'échelle**



f(t) f(2t)

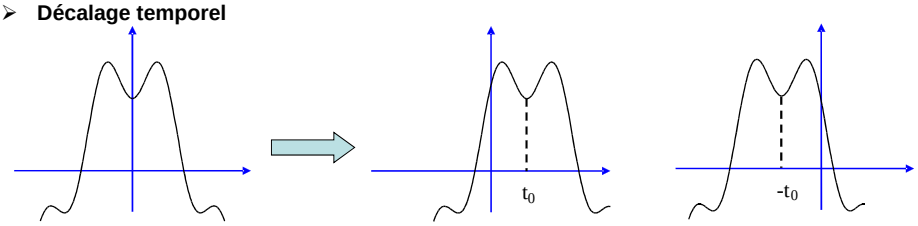


f[n] f[2n]

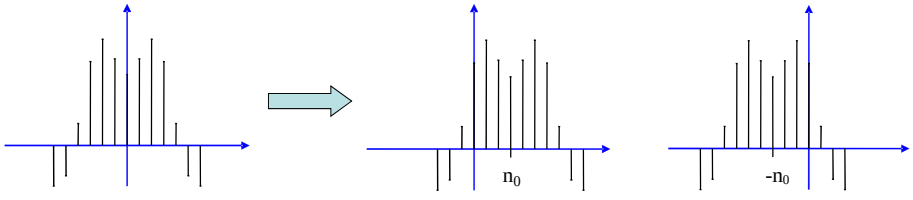
TRANSFORMATIONS DE LA VARIABLE INDEPENDANTE

34

- **Décalage temporel**



f(t) f(t-t_0) → Retard f(t+t_0) → Avance

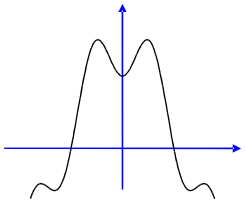


f[n] f[n-n_0] → Retard f[n+n_0] → Avance

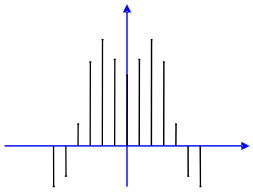
TRANSFORMATIONS DE LA VARIABLE INDEPENDANTE

35

- **Signaux pairs : invariance par retournement**

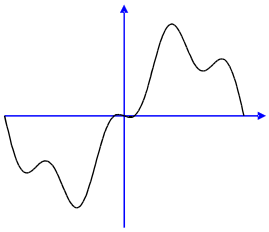


f(-t) = f(t)

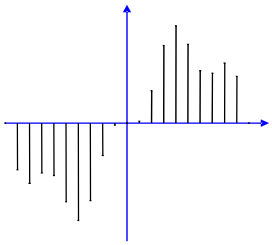


f[-n] = f[n]

- **Signaux impairs : symétrie par retournement**



f(-t) = - f(t)

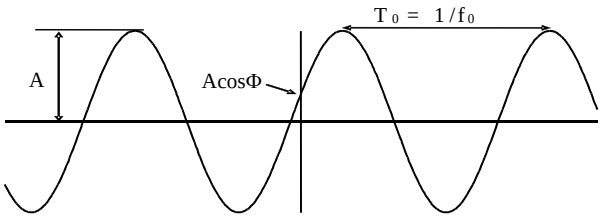


f[-n] = - f[n]

SIGNAUX DE BASE CONTINUS

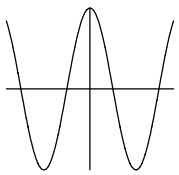
36

- **Signal sinusoïdal :** $f(t) = A\cos(2\pi f_0 t + \phi)$

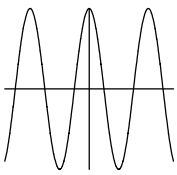


A : amplitude ϕ : phase f_0 : fréquence T_0 : période

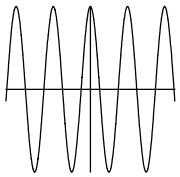
- **Variation de la fréquence f_0 :**



f_1



f_2



f_3

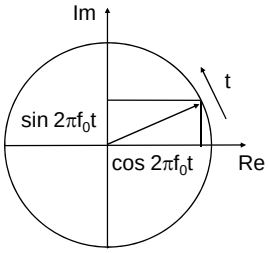
$f_1 < f_2 < f_3$

SIGNAUX DE BASE CONTINUS

➢ **Exponentielle complexe** $e^{j2\pi f_0 t} = \cos 2\pi f_0 t + j \sin 2\pi f_0 t$

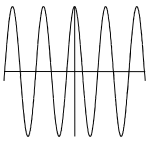
$$\begin{cases} \mathrm{Re}(e^{j2\pi f_0 t}) = \cos 2\pi f_0 t \\ \mathrm{Im}(e^{j2\pi f_0 t}) = \sin 2\pi f_0 t \end{cases} \qquad \begin{cases} \cos 2\pi f_0 t = \frac{1}{2}(e^{j2\pi f_0 t} + e^{-j2\pi f_0 t}) \\ \sin 2\pi f_0 t = \frac{1}{2j}(e^{j2\pi f_0 t} - e^{-j2\pi f_0 t}) \end{cases}$$

➢ **Interprétation**

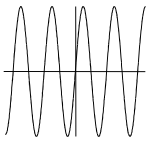


f_0: vitesse de rotation

Partie réelle



Partie imaginaire



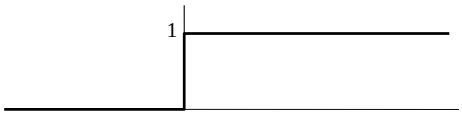
➢ **Exponentielles harmoniques** $f_k(t) = e^{j2\pi k f_0 t}, k \in Z$

f_0 : fréquence fondamentale

SIGNAUX DE BASE CONTINUS

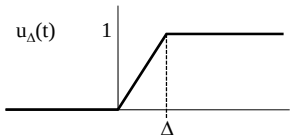
➢ **Échelon unité**

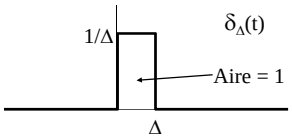
$$u(t) = \begin{cases} 0 \text{ si } t < 0 \\ 1 \text{ si } t > 0 \end{cases}$$



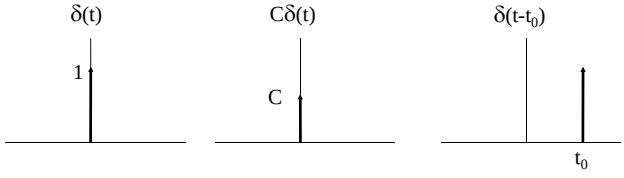
➢ **Impulsion unité (ou Dirac) : δ(t)** on veut $u(t) = \int_{-\infty}^{t} \delta(\tau) d\tau$ à savoir $\delta(t) = \frac{du(t)}{dt}$

$\delta(t) = \lim_{\Delta \to 0} \delta_\Delta(t)$

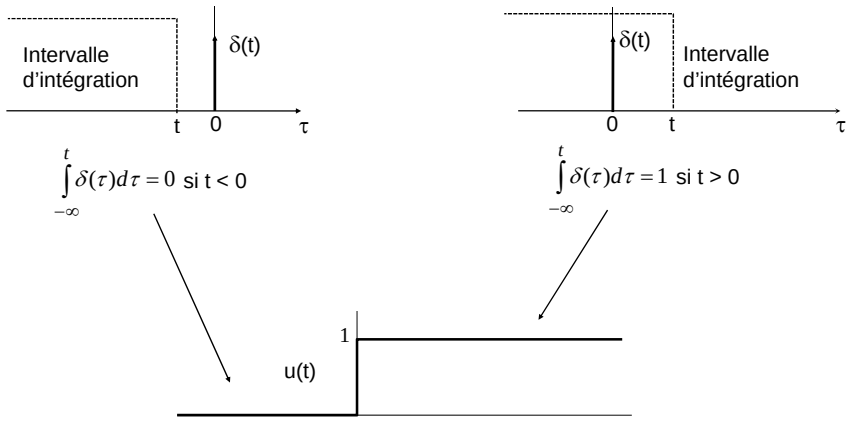




➔ **Représentation**



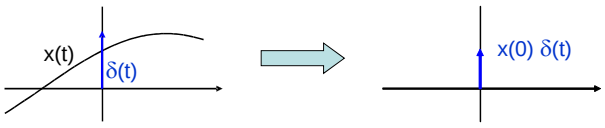
➢ **Dirac : interprétation**



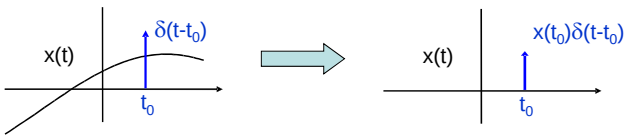
→ L'aire du Dirac est "concentrée" en un point

➢ **Dirac : Propriété** $\int_{-\infty}^{+\infty} \delta(t)dt = 1$

➢ **Dirac : Propriété** x(t) δ(t) = x(0) δ(t) → Dirac de poids x(0)



➢ **Dirac : Propriété** x(t) $\delta(t-t_0)$ = $x(t_0)$ $\delta(t-t_0)$



➢ **Dirac : Propriété** $\int_{-\infty}^{+\infty} x(t)\delta(t)dt = x(0)$

➢ **Dirac : Propriété** $\int_{-\infty}^{+\infty} x(t)\delta(t-t_0)dt = x(t_0)$

➢ Énergie d'un signal continu s(t)

- Énergie moyenne calculée sur l'intervalle [t_1, t_2] : $E_s(t_1,t_2) = \int_{t_1}^{t_2} |s(t)|^2 \, dt$

- Énergie **totale** : $E_s = \int_{-\infty}^{+\infty} |s(t)|^2 \, dt$

➢ **Signaux à énergie finie :** $E_s < \infty$

➢ **Exemples : les signaux suivants sont-ils à énergie finie ?**
- u(t)
- $u(t)e^{-t/\tau}$
- Sinusoïde
- Signaux périodiques

SIGNAUX DE BASE CONTINUS

➢ Puissance d'un signal continu s(t)

- Puissance moyenne calculée sur l'intervalle $[t_1, t_2]$:

  → **Interprétation : énergie dissipée par unité de temps**

$$P_s(t_1,t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |s(t)|^2 \, dt$$

- Puissance **totale,** cas général :

$$P_s = \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} |s(t)|^2 \, dt$$

- Puissance **totale**, cas des signaux périodiques

$$P_s = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} |s(t)|^2 \, dt$$

➢ **Signaux à puissance finie :** $P_s < \infty$

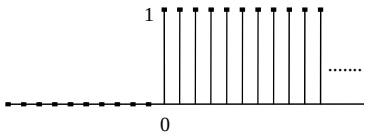
➢ **Exemples : les signaux suivants sont-ils à puissance finie ?**

- u(t), $u(t)e^{-t/\tau}$
- Cas général des signaux à énergie finie

SIGNAUX DE BASE DISCRETS

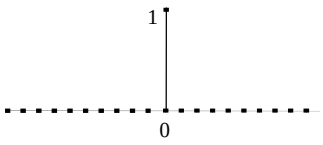
➢ **Échelon unité**

$$u[n] = \begin{cases} 0 \text{ si } n < 0 \\ 1 \text{ si } n \geq 0 \end{cases}$$



➢ **Impulsion unité : δ[n]** on veut $u[n] = \sum_{m=-\infty}^{n} \delta[n]$ à savoir $\delta[n] = u[n] - u[n-1]$

$$\delta[n] = \begin{cases} 1 \text{ si } n = 0 \\ 0 \text{ si } n \neq 0 \end{cases}$$



➢ **Propriétés**

$$\sum_{n=-\infty}^{+\infty} \delta[n] = 1$$

$$\sum_{n=-\infty}^{+\infty} x[n]\delta[n] = x[0]$$

$x[n]\,\delta[n] = x[0]\,\delta[n]$

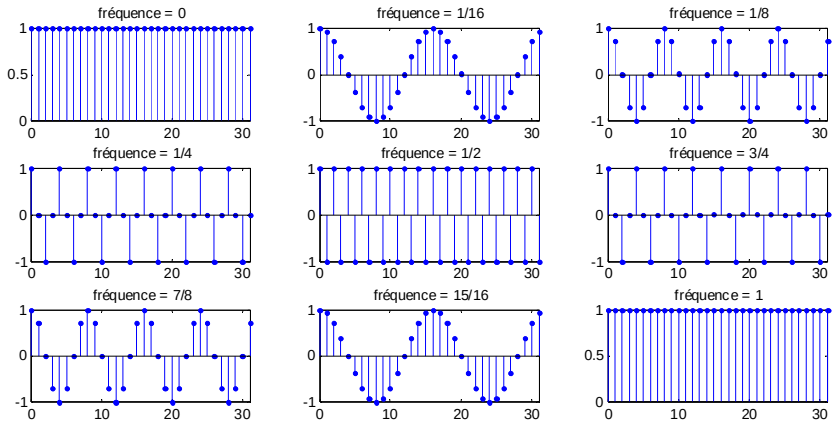
$$\sum_{n=-\infty}^{+\infty} x[n]\delta[n - n_0] = x[n_0]$$

$x[n]\,\delta[n-n_0] = x[n_0]\,\delta[n-n_0]$

SIGNAUX DE BASE DISCRETS

➢ **Sinusoïdes discrètes : cos $2\pi f_0 n$**

- A la différence du cas continu, la rapidité des oscillations n'augmente pas continûment avec f_0



SIGNAUX DE BASE DISCRETS

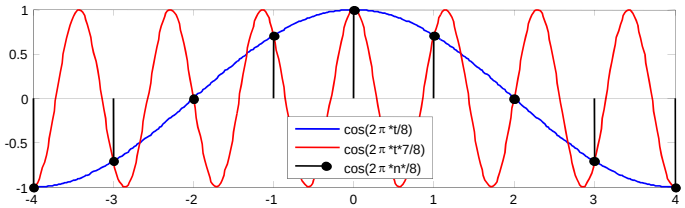
➢ **Sinusoïdes discrètes : cos $2\pi f_0 n$**

- A la différence du cas continu, la rapidité des oscillations n'augmente pas continûment avec f_0

- <u>en continu</u> : $\cos 2\pi f_1 t \neq \cos 2\pi f_2 t$ si $f_2 \neq f_1$

  <u>en discret</u> : $\cos 2\pi f_1 n = \cos 2\pi f_2 n$ si $f_2 = f_1 \pm k$, $k \in N$

Exemple avec
- $f_1 = 1/8$
- $f_2 = 1/8\text{-}1 = 7/8$



➢ **Sinusoïdes discrètes**

💣 Périodicité

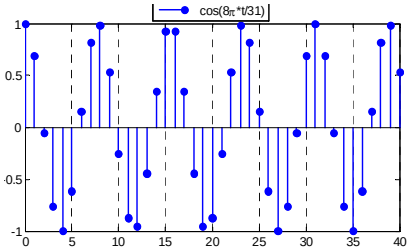
cos $2\pi f_0(n+N)$ = cos $2\pi f_0(n)$ → N = k/f_0

Pour que N soit un entier, il faut donc que $f_0 = k/N$ rationnel

Exemples

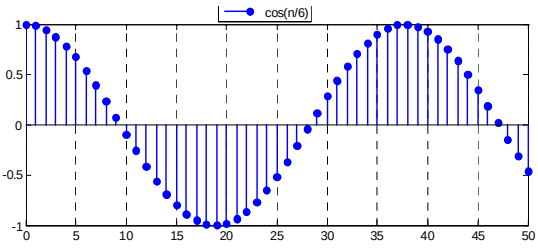
$\cos(\frac{8\pi}{31}n)$ → $f_0 = 4/31$ (rationnel)

Période N= k*31/4 → N = 31 (avec k=4)



cos n/6 $f_0 = 1/12\pi$, (non rationnel)

Période N = k * 12π → Non périodique



➢ **Exponentielles discrètes**

$$e^{j2\pi f_0 n} = \cos 2\pi f_0 n + j \sin 2\pi f_0 n$$

💣 **Mêmes propriétés que sinusoïdes (différence de comportement par rapport au cas continu et contraintes sur la périodicité)**

➢ **Énergie et puissance d'un signal discret s[n]**

Énergie	
Énergie moyenne	$\sum_{n=n_1}^{n_2} \lvert s[n] \rvert^2$
Énergie totale E_s	$\sum_{n=-\infty}^{+\infty} \lvert s[n] \rvert^2$
Condition pour énergie finie	$E_s < \infty$

Puissance	
Puissance moyenne	$\frac{1}{n_2 - n_1} \sum_{n=n_1}^{n_2} \lvert s[n] \rvert^2$
Puissance totale P_s (cas général)	$\lim_{N \to \infty} \frac{1}{N} \sum_{n=-N/2}^{N/2} \lvert s[n] \rvert^2$
Puissance totale P_s (signaux périodiques)	$\frac{1}{N_0} \sum_{n=-N_0/2}^{N_0/2} \lvert s[n] \rvert^2$
Condition pour puissance finie	$P_s < \infty$

Les Systèmes Linéaires Invariants

Notion de Système

51

- **Définition :**
  - Un système est un modèle mathématique d'un processus physique qui relie un signal d'entrée à un signal de sortie
  - Un système est un dispositif de traitement du signal
  - En entrée : e(t) signal d'entrée
  - En sortie : s(t) signal de sortie

- **Exemples :**
  - Amplificateur, système audio, téléphone, système vidéo
  - Un système complexe peut être vu comme l'interconnexion de plusieurs systèmes dont les fonctions sont plus simples

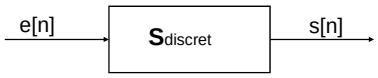
- **Questions**
  - Comment caractériser un système ?
  - Quelles sont les propriétés intéressantes des systèmes ?
  - Comment modéliser la relation entre entrée et sortie ?

Représentation des Systèmes

52

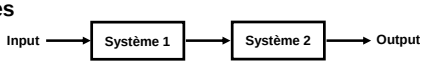
- **Représentation sous forme de schéma bloc**

s(t) = S{e(t)}

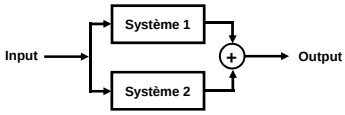


s[n] = S{e[n]}

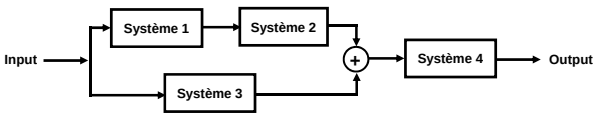
- **Interconnections des systèmes**
  - Série / Cascade



  - Parallèle



  - Série / Parallèle



  - Feed-back

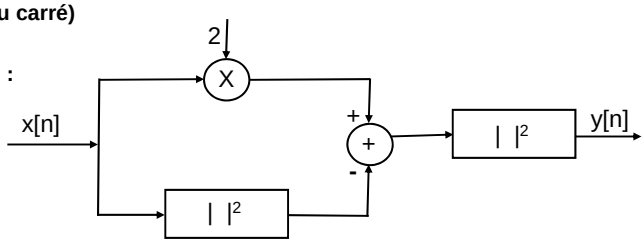
EXEMPLE DE SYSTEMES

AMPLIFICATION
RETARD
MOYENNE GLISSANTE
FILTRAGE

MODULATION
DETECTION
REDRESSEUR (valeur absolue)
QUADRATEUR (élever au carré)

- **Réalisation de systèmes :**

$Y[n] = (2x[n] - x[n]^2)^2$



Corrélateur $C_{xy}(\tau) = \int_{\tau - T/2}^{\tau + T/2} x(t-\tau)y(t)dt$

PROPRIETES DES SYSTEMES

ETUDE DE 2 PROPRIETES FONDAMENTALES (système continu ou discret):
LINEARITE
INVARIANCE EN TEMPS

LINEARITE (Propriété de superposition)

Soit $y_1[n]=S\{x_1[n]\}$ et $y_2(t)=S\{x_2[n]\}$ ALORS $a\,y_1[n]+ b\,y_2[n] = S\{a\,x_1[n]+b\,x_2[n]\}$

Conséquence : une entrée nulle produit une sortie nulle

INVARIANCE EN TEMPS

Soit $y[n]=S\{x[n]\}$ ALORS $y[n-n_0]=S\{x[n-n_0]\}$
La sortie du système ne dépend pas de l'origine des temps
La sortie du système ne dépend pas de l'instant où est appliquée l'entrée

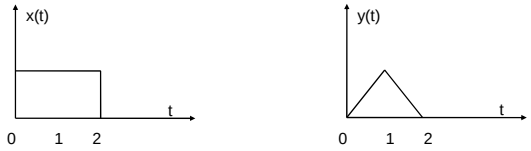
On notera SLIT un système **L**inéaire et **I**nvariant en **T**emps

➢EXEMPLES - linéarité et invariance en temps des systèmes suivants :

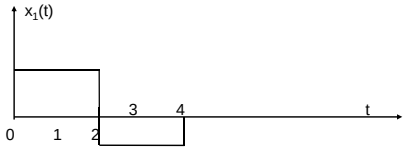
y[n] = 2x[n], y(t) = x(t-2) -2x(t-19) LIT
y[n] = x[-n], y(t) = sin(6t) x(t) L
y[n] = 2x[n]+3, y(t) = a exp(x(t)) IT

➢UTILISATION DES PROPRIETES :

Soit un système LINEAIRE et INVARIANT EN TEMPS
qui a en sortie y(t) lorsque l'entrée est x(t) :



Quelle est la réponse du système lorsque l'entrée est $x_1(t)$



RAPPEL DE QUELQUES PROPRIETES DE L'IMPULSION UNITE

$$x[n]\delta[n] = x[0]\delta[n] \qquad x(t)\delta(t) = x(0)\delta(t)$$

$$x[n]\delta[n-n_0] = x[n_0]\delta[n-n_0] \qquad x(t)\delta(t-t_0) = x(t_0)\delta(t-t_0)$$

$$\sum_{-\infty}^{+\infty}\delta[n] = 1 \qquad \int_{-\infty}^{+\infty}\delta(t)dt = 1$$

$$\sum_{-\infty}^{+\infty}x[n]\delta[n] = x[0] \qquad \int_{-\infty}^{+\infty}x(t)\delta(t)dt = x(0)$$

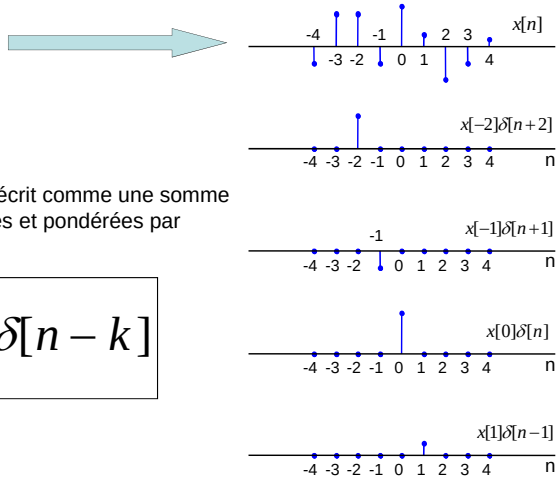
$$\sum_{-\infty}^{+\infty}x[n]\delta[n-n_0] = x[n_0] \qquad \int_{-\infty}^{+\infty}x(t)\delta(t-t_0)dt = x(t_0)$$

$$\sum_{k=-\infty}^{n}\delta[k] = u[n] \qquad \int_{-\infty}^{t}\delta(\tau)d\tau = u(t)$$

REPRESENTATION DES SIGNAUX EN SOMME D'IMPULSIONS

57

- **Exemple de la représentation d'un signal discret en terme d'impulsions retardées**



- **Généralisation :**
  - Tout signal discret peut être décrit comme une somme d'impulsions de Dirac retardées et pondérées par l'amplitude de ce signal :

$$x[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k]$$

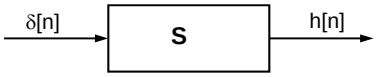
- **Décomposition équivalente pour les signaux continus :**

$$x(t) = \int_{-\infty}^{+\infty} x(\tau)\delta(t-\tau)d\tau$$

NOTION DE REPONSE IMPULSIONNELLE

58

- **La réponse impulsionnelle h[n] d'un système est la réponse du système lorsque le signal d'entrée est l'impulsion de Dirac δ[n] :**



- **Dans le cas des systèmes linéaires et invariant en temps (SLIT), la réponse impulsionnelle h[n] permet de caractériser totalement le système.**

  La Transformée de Fourier ou la transformée en z de la réponse impulsionnelle permettent de déduire :
  - La réponse en fréquence ou gain complexe H(f)
  - Le gain en fonction de fréquence |H(f)| ou la phase arg(H(f)
  - La fonction de transfert H(z)

EQUATION DE CONVOLUTION

- **Considérons un système S linéaire et invariant en temps (LTI) de réponse impulsionnelle h[n]. Nous pouvons donc écrire :**

entrée	S	sortie
δ[n]		h[n]
δ[n-k]		h[n-k]
x[k] δ[n-k]		x[k] h[n-k]
$\sum_{k=-\infty}^{+\infty} x[k]\,\delta[n-k]$		$\sum_{k=-\infty}^{+\infty} x[k]\,h[n-k]$
x[n]	S	y[n]

- **D'où l'équation de convolution discrète qui lie signal d'entrée, signal de sortie et réponse impulsionnelle d'un système LIT :**

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]\,h[n-k]$$

EQUATION DE CONVOLUTION

- **Un changement de variable permet de convertir l'équation de convolution sous une autre forme:**

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]\,h[n-k] = \sum_{k=-\infty}^{+\infty} h[k]\,x[n-k]$$

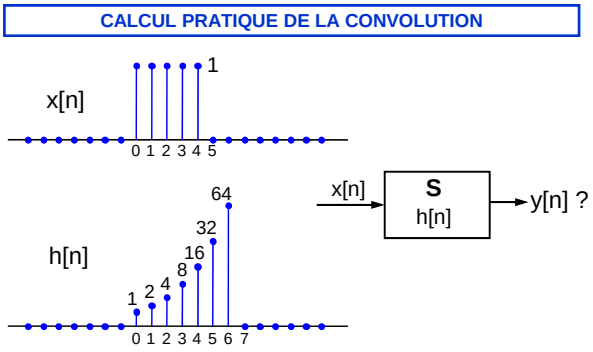
- **Notation de la convolution**

$$y[n] = x[n] * h[n] = h[n] * x[n]$$

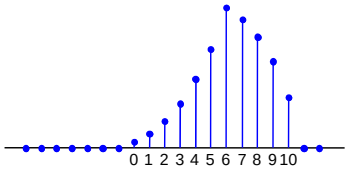
- **L'équation de convolution permet de calculer la sortie du système pour n'importe quelle entrée x[n]**

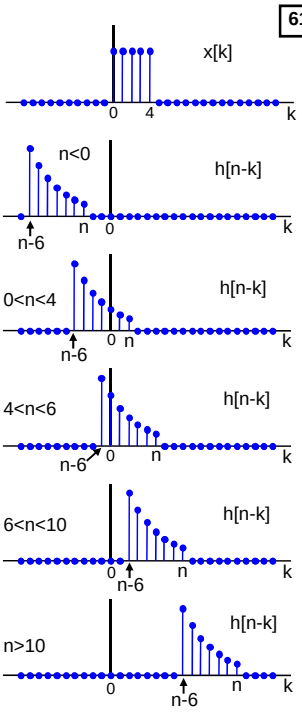
CALCUL PRATIQUE DE LA CONVOLUTION

61



y[n] = x[n] * h[n]





CALCUL PRATIQUE DE LA CONVOLUTION

62

- **Cas usuel :**
  - soit x de durée N, h de durée M, avec N≥M (signaux à durée finie)
  - x et h ont des amplitudes nulles pour les temps (indices) négatifs (signaux causaux)

  alors l'équation de convolution se simplifie

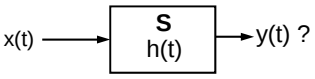
$$y[n] = \sum_{k=0}^{N-1} x[k]\, h[n-k] = \sum_{k=0}^{M-1} h[k]\, x[n-k] \;\; pour \; n = 0 \; à \; N+M$$

- **On note que la durée de y[n] est N+M-1 : la convolution 'allonge' les signaux**

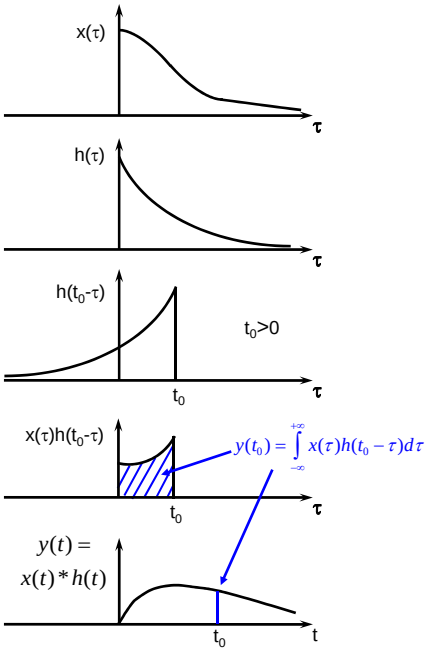
$$y[0] = x[0]\, h[0]$$
$$y[1] = x[0]\, h[1] + x[1]\, h[0]$$
$$y[2] = x[0]\, h[2] + x[1]\, h[1] + x[2]\, h[0]$$
$$.....$$
$$y[i] = x[i-M+1]\, h[M-1] + ... + x[i-M+j]\, h[M-j] + ... + x[i]\, h[0]$$
$$.....$$
$$y[N+M] = x[N]\, h[M-1]$$

ILLUSTRATION DE LA CONVOLUTION DE SIGNAUX CONTINUS

63



$$y(t) = x(t) * h(t)$$
$$= \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau$$
$$= \int_{-\infty}^{+\infty} h(\tau)x(t-\tau)d\tau$$



PROPRIETES DE LA CONVOLUTION

64

- **Commutativité**

$$x[n] * h[n] = h[n] * x[n]$$

- **Mise en cascade**

$$x[n] * \{h_1[n] * h_2[n]\} = \{x[n] * h_1[n]\} * h_2[n]$$

- **Distributivité**

$$x[n] * \{h_1[n] + h_2[n]\} = \{x[n] * h_1[n]\} + \{x[n] * h_2[n]\}$$

- **Élément neutre, définition de la réponse impulsionnelle d'un système LIT**

$$\delta[n] * h[n] = h[n]$$

- **Attention, ces propriétés ne sont valables que pour les systèmes LIT**

PROPRIETES DE LA CONVOLUTION

- **Fonction propre d'un système LIT**

Soit un système LIT de réponse impulsionnelle h(t)

Considérons le signal d'entrée x(t) exponentielle complexe (phaseur pur) :

$$x_{f_0}(t) = e^{2\pi j f_0 t}$$

Calculons la sortie correspondante à ce système:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} h(\tau) e^{2\pi j f_0 (t-\tau)} d\tau$$

$$= e^{2\pi j f_0 t} \int_{-\infty}^{+\infty} h(\tau) e^{-2\pi j f_0 \tau} d\tau = e^{2\pi j f_0 t} H(f_0)$$

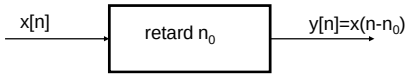
$H(f_0)$ est une constante complexe dont la valeur ne dépend que de f_0 et de la réponse impulsionnelle du système

Résultat extrêmement important : La réponse d'un système LIT à un signal d'entrée de type exponentielle complexe (fonction sinusoïdale de fréquence fixée), est le même signal (pas de changement de fréquence) mais pondéré par un coefficient qui dépend uniquement de la réponse impulsionnelle du système.

On dit que l'exponentielle complexe est une fonction propre des systèmes SLIT

EXEMPLES DE SYSTEMES LIT

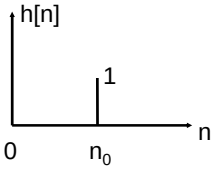
- **RETARD PUR**



$$y[n] = x[n - n_0] = x[n] * \delta[n - n_0]$$

- **On déduit que la réponse impulsionnelle de ce système est**

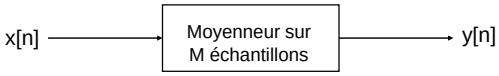
$h[n]=\delta[n-n_0]$



EXEMPLES DE SYSTEMES LIT

67

- **MOYENNEUR TEMPOREL**

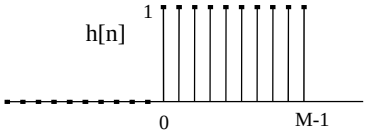


$$y[n] = x[n] + x[n-1] + x[n-2] + ... + x[n-M+1]$$

$$= \sum_{k=0}^{k=M-1} x[n-k] = \sum_{k=0}^{k=M-1} x[n] * \delta[n-k] = x[n] * \sum_{k=0}^{k=M-1} \delta[n-k]$$

- **On déduit que la réponse impulsionnelle de ce système est**

$$h[n] = \sum_{k=0}^{k=M-1} \delta[n-k]$$



PROPRIETES DES SYSTEMES

68

- **LINEARITE**
- **INVARIANCE EN TEMPS**
- **CAUSALITE**

  Définition générale : La sortie d'un système causal ne dépend que des valeurs présentes ou passées de l'entrée

  Cas d'un système LIT : un système LIT est causal si sa réponse impulsionnelle h[n] est nulle pour n≤0

- **STABILITE**

  Définition générale: Un système est stable si à toute entrée d'amplitude bornée correspond une sortie bornée

  Cas d'un système LIT : un système LIT est stable si et seulement si : $\sum_{k=-\infty}^{+\infty} |h[k]| < \infty$

- **INVERSIBILITE**

  Un système LIT de réponse impulsionnelle h[n] est inversible si on peut trouver une réponse impulsionnelle inverse h1[n] telle que

  $h[n] * h1[n] = \delta[n]$

- **Les caractéristiques des LIT sont étudiées avec la Transformée en Z**

- **Réponse indicielle s[n]:**

  Un système peut être caractérisé par sa réponse à l'échelon u[n].

  La réponse impulsionnelle peut être retrouvée par dérivation de la réponse indicielle

DESCRIPTION DES SYSTEMES EQUATIONS AUX DIFFERENCES

69

- **Un sous-ensemble très important de systèmes discrets LIT sont ceux dont l'entrée x[n] et la sortie y[n] sont reliés par une équation aux différences à coefficients constants**
  (de même les systèmes continus qui sont décrits par des équations différentielles)

- **La représentation par équations aux différences d'un système LIT est très importante parce qu'elle permet d'envisager de manière efficace la réalisation de ces systèmes**

- **La représentation par équations aux différences permet de connaître les caractéristiques du système (ordre du système, fréquences transmises ou atténuées…)**

- **La Transformée en Z est une outil très efficace pour la réalisation et l'analyse des caractéristiques de ces systèmes**

DESCRIPTION DES SYSTEMES EQUATIONS AUX DIFFERENCES

70

- **L'expression la plus générale reliant entrée x[n] et sortie y[n] d'un système LIT est de la forme :**

$$\sum_{k=-\infty}^{+\infty} b_k x[n-k] = \sum_{k=-\infty}^{+\infty} a_k y[n-k]$$

- **Si on considère un système causal d'ordre N, l'expression peut s'écrire :**

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$$

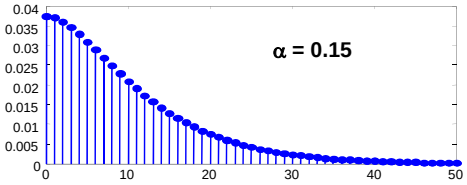
- **Exemple d'un système du 1er ordre : y[n]=x[n] +Ky[n-1]**

- **Exemple d'un système du 2ème ordre : y[n]=x[n] +a1y[n-1]+ a2y[n-2]**

DESCRIPTION DES SYSTEMES EQUATIONS AUX DIFFERENCES

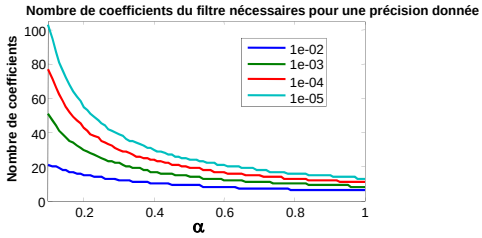
- **Exemple : soit le filtre causal de réponse impulsionnelle h[n]**

$$h[n] = k(\alpha n + 1)e^{-\alpha n}$$



- **Filtre à réponse impulsionnelle infinie (IIR) : pas d'implantation exacte par convolution**

- **Solution 1 : troncature de la réponse impulsionnelle pour une précision donnée**



DESCRIPTION DES SYSTEMES EQUATIONS AUX DIFFERENCES

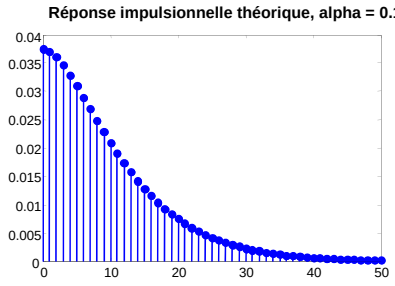
- **Solution 2 : Transformée en z → réalisation récursive par équation aux différences**

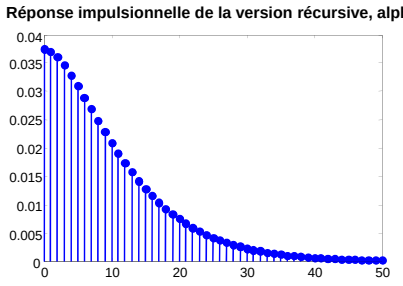
$$y[n] = b_0 x[n] + b_1 x[n-1] - a_1 y[n-1] - a_2 y[n-2]$$

$$y[n] = k\left(x[n] + e^{-\alpha}(\alpha - 1)x[n-1]\right) + 2e^{-\alpha} y[n-1] - e^{-2\alpha} y[n-2]$$

→ Complexité: 4 multiplications et 3 additions, quel que soit α

- **Comparaison des 2 implantations**





➢ **Vérification : calcul de la réponse impulsionnelle de la solution récursive**

$$y[n] = k\left(x[n] + e^{-\alpha}(\alpha - 1)x[n-1]\right) + 2e^{-\alpha}y[n-1] - e^{-2\alpha}y[n-2] \qquad \text{avec } x[n] = \delta[n]$$

$$y[0] = k$$

$$y[1] = ke^{-\alpha}(\alpha - 1) + 2e^{-\alpha}y[0] = ke^{-\alpha}(\alpha - 1) + 2e^{-\alpha}k = k(\alpha + 1)e^{-\alpha}$$

$$y[2] = 2e^{-\alpha}y[1] - e^{-2\alpha}y[0] = k(2\alpha + 1)e^{-2\alpha}$$

$$y[3] = 2e^{-\alpha}y[2] - e^{-2\alpha}y[1] = k(3\alpha + 1)e^{-3\alpha}$$

➢ **Terme général**

On suppose $y[n] = k(\alpha n + 1)e^{-\alpha n}$

On vérifie alors

$$y[n+1] = 2e^{-\alpha}y[n] - e^{-2\alpha}y[n-1]$$

$$= 2e^{-\alpha}k(\alpha n + 1)e^{-\alpha n} - e^{-2\alpha}k\left[\alpha(n-1) + 1\right]e^{-\alpha(n-1)}$$

$$= ke^{-\alpha(n+1)}(2\alpha n + 2 - \alpha n + \alpha - 1)$$

$$= k\left[\alpha(n+1) + 1\right]e^{-\alpha(n+1)}$$

cqfd

QUATRE THEOREMES FONDAMENTAUX

• Théorème de Parceval

Considérons un signal d'énergie finie. Cette énergie vaut : $W = \int_{-\infty}^{+\infty} x^2(t)dt$

Sa Densité Spectrale d'Energie vaut : $S_x(f) = |X(f)|^2$

La relation suivante constitue le théorème de Parceval : $\boxed{\int_{-\infty}^{+\infty} x^2(t)dt = \int_{-\infty}^{+\infty} |X(f)|^2 df}$

Compte tenu de la relation liant les signaux en entrée et sortie des systèmes linéaire, on déduit la relation entre leur Densité Spectrale d'Energie : $S_y(f) = |H(f)|^2 . S_x(f)$

• Théorème du retard

$x(t-t_0)$ s'écrit également $x(t) \otimes \delta(t-t_0)$, a pour Transformée de Fourier $X(f)e^{-j2\pi ft_0}$

Le décalage temporel se traduit par un déphasage de le TF, il n'affecte pas le module de la TF.
Cela est la conséquence du choix de la décomposition sur une base d'exponentielles complexes.

• Théorème du produit de convolution

$x(t) \otimes y(t)$ a pour Transformée de Fourier $X(f).Y(f)$
et réciproquement $X(f) \otimes Y(f)$ a pour TF $x(t).y(t)$

• Théorème de Wiener Kintchine

Soit la fonction d'autocorrélation d'un signal déterministe d'énergie finie:

$$\boxed{R_x(\tau) = \int_{-\infty}^{+\infty} x(t)x(t-\tau)dt}$$

La relation suivante constitue le théorème de Wiener Kintchine :

$$\boxed{S_x(f) = \int_{-\infty}^{+\infty} R_x(\tau)e^{-j2\pi f\tau}d\tau}$$

Ainsi, la Densité Spectrale d'Energie est la transformée de Fourier de la fonction d'autocorrélation.
Ce théorème est valable pour les signaux à énergie finie et subsiste pour les signaux aléatoires.

Ouverture Thématique

Signal

Signal & Fourier : du temps aux fréquences

Hugues BENOIT-CATTIN

Département Télécommunications, Services & Usages

Séance 2 et 3 - **Fourier - un outil d'analyse fréquentielle** - Historique, définitions - Propriétés - Etudes de cas : Le bruit en temps et en fréquence, signaux aléatoire, Le signal de parole - Exercice calcul TF en continu - Théorème d'échantillonnage - Transformée de Fourier discrète, relation avec le calcul pratique de la FFT et des conventions d'écriture de la fréquence normalisée

1807 : Equations de la chaleur
>> Equa. Diff
>> Séries de Fourier
>> Résoudre l'équation

Né en 1768, Joseph Fourier se révèle très tôt doué pour les lettres et les sciences. Mais c'est l'étude des mathématiques qui provoque chez lui enthousiasme et passion. En 1789, il viendra à Paris, devant l'académie, lire son premier mémoire sur les équations algébriques. Joseph Fourier va ensuite enseigner à Auxerre puis à l'école polytechnique de Paris. Il participera à l'expédition d'Egypte et sera chargé à son retour en France, d'écrire la préface historique de l'ouvrage, qui regroupe l'ensemble des observations faites au cours de l'expédition. C'est en 1802 que Fourier est nommé Préfet de l'Isère. Grâce à sa puissance de travail, il réalisera au cours de son mandat de grands travaux : assèchement des marais de Bourgoin, tracé de la route de Grenoble à Turin. Il prêtera aussi une grande attention à l'enseignement mis en place dans les lycées (1804) et à la faculté des Sciences (1811). Et c'est aussi pendant son mandat de préfet qu'il effectue l'essentiel de son oeuvre scientifique. De retour à Paris, il entrera à l'Académie en 1816, tout en continuant ses travaux de recherche concernant la propagation de la chaleur, les températures du globe terrestre et des espaces planétaires.
En 1826, il entre à l'Académie Française et, malgré sa maladie, travaillera inlassablement jusqu'à la fin de sa vie, le 17 Mai 1830.

Séries de Fourier

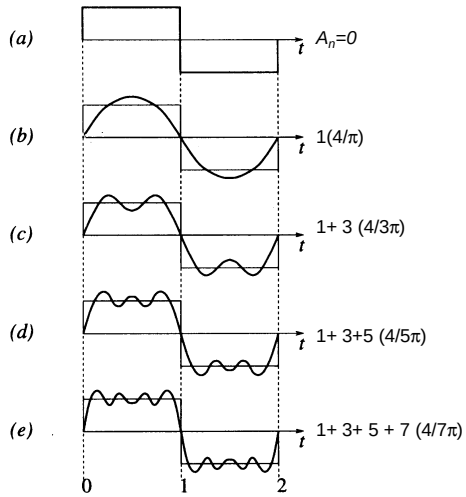
Tout signal périodique (T) de puissance finie peut être décomposé en une somme de sinus et de cosinus.

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos(nw_0 t) + B_n \sin(nw_0 t)$$

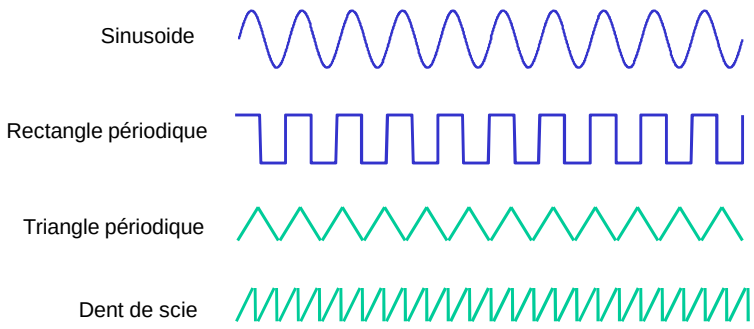
$$A_n = \frac{2}{T} \int_{-T/2}^{T/2} x(t).\cos(nw_0 t)dt \qquad n = 0,1,2,...$$

$$B_n = \frac{2}{T} \int_{-T/2}^{T/2} x(t).\sin(nw_0 t)dt \qquad n = 1,2,3...$$

Continu / Fondamental / harmoniques



Quelques exemples de signaux périodiques



$$x(t) = \{A_n : n = 0,1,2,...\} + \{B_n : n = 1,2,3...\}$$

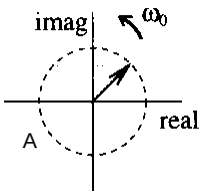
Séries de Fourier complexe

De $\{A_n,B_n\}$ à X_n en utilisant la notation complexe

$$A\exp(jw_0t) = A\cos(w_0t) + jA\sin(w_0t)$$

$$x(t) = \sum_{n=-\infty}^{+\infty} X_n \exp(jnw_0t)$$

$$X_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\exp(jnw_0t)dt$$

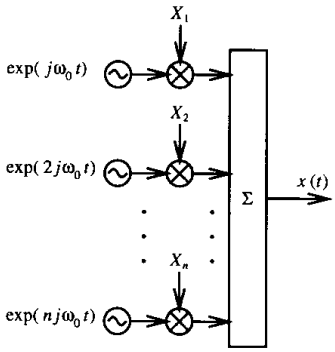


$$f_0 = 1/T$$

$$w_0 = 2\pi f_0$$

$$A\cos(w_0t) = \mathrm{Re}\{A\exp(jw_0t)\}$$

$$A\sin(w_0t) = \mathrm{Im}\{A\exp(jw_0t)\}$$

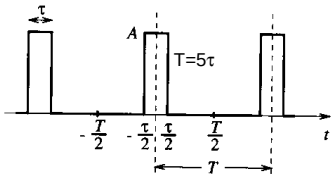


$$\cos(nw_0t) = \frac{\exp(jnw_0t) + \exp(-jnw_0t)}{2}$$

$$\sin(nw_0t) = \frac{\exp(jnw_0t) - \exp(-jnw_0t)}{2j}$$

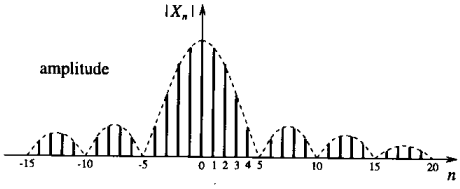
Pour un cosinus ?
Pour un sinus ?

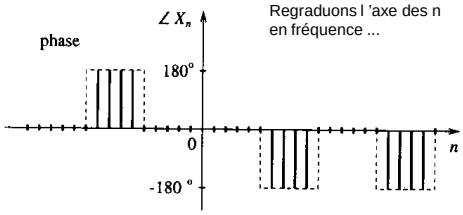
Pour ça ?



$$X_n = \frac{A\tau}{T}\frac{\sin(nw_0\tau/2)}{nw_0\tau/2}$$

Le spectre du signal





Regraduons l 'axe des n en fréquence ...

Transformée de Fourier des signaux continus périodiques

$$x(t) = f_0 \sum_{f \in Z(f_0)} X(f) e^{j2\pi f t} \qquad t \in R, \text{ et } Z(f_0) = \{..., -2f_0, -f_0, 0, f_0, 2f_0, ...\}$$

$$X(f) = \int_{-T/2}^{T/2} x(t) e^{-j2\pi f t} dt \qquad f \in Z(f_0), t \in R$$

Signal périodique ⇔ Spectre discret

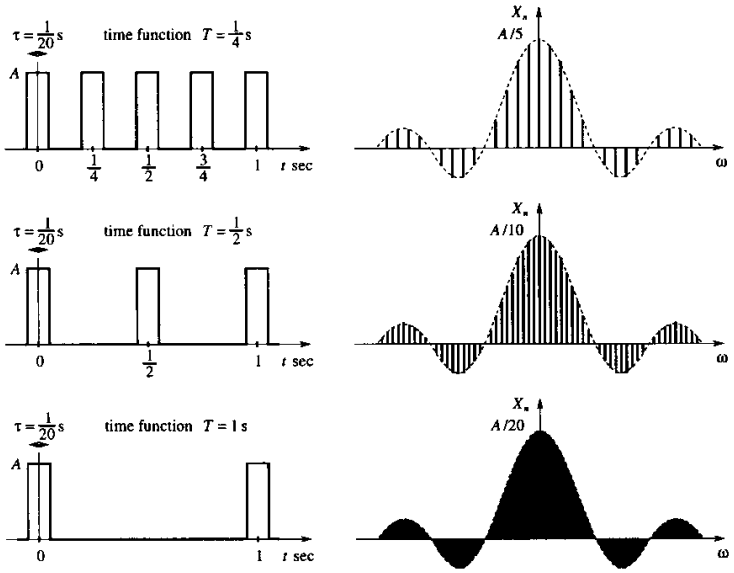
Et la puissance d 'un signal périodique ?

$$P = \frac{1}{T}\int_0^T x^2(t).dt$$

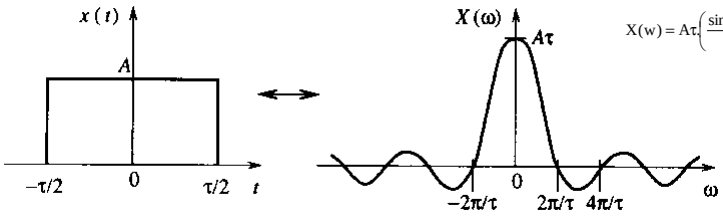
$$P = \frac{A_0^2}{4} + \frac{1}{2}\sum_{n=1}^{\infty}\left(A_n^2 + B_n^2\right)$$ Identité de Parseval

$$P = \sum_{n=-\infty}^{\infty}\left|X_n\right|^2$$ Densité Spectrale de Puissance

Du périodique au non-périodique



Transformée de Fourier des signaux continus



Regraduons en f

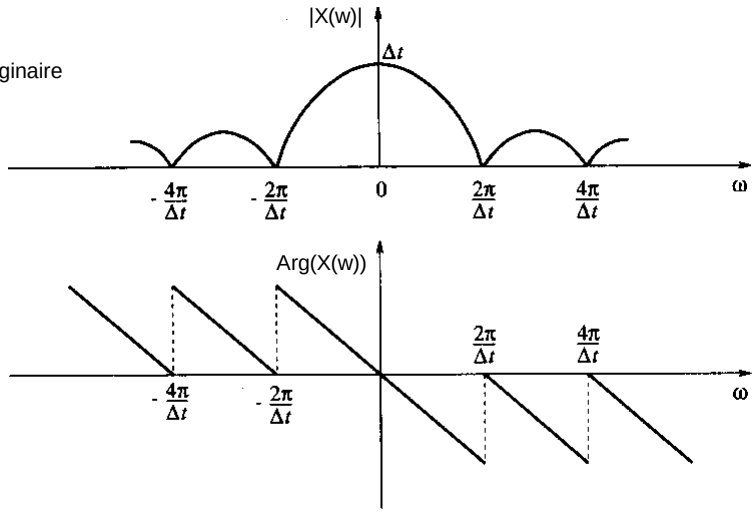
$$X(f) = \int_{-\infty}^{+\infty} x(t)\exp(-j2\pi f\, t)dt \quad ,t \in R, f \in R$$

$$x(t) = \int_{-\infty}^{+\infty} X(f)\exp(j2\pi f\, t)df \quad ,t \in R, f \in R$$

Représentation de la TF

Module / Argument

Parties réelle & imaginaire



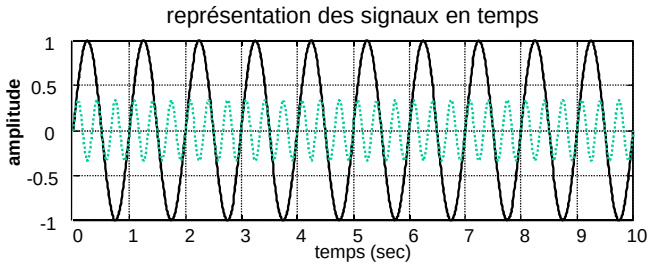
Quelques propriétés de Transformée de Fourier

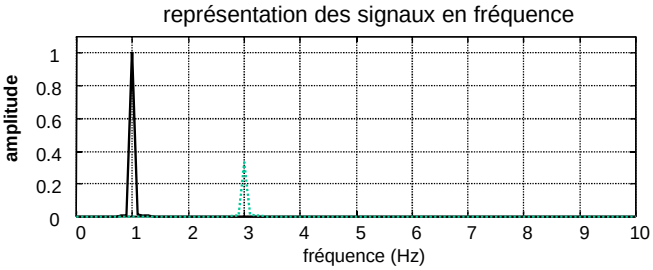
- Linéarité
- $X(f)$ ↖ module $|X(f)|$, phase $Arg[X(f)]$
- $x(t)$ réel ⇔ $Re[X(f)]$ paire, $Im[X(f)]$ impaire, module pair, phase impaire
- $x(t)$ réel pair ⇔ X(f) réel pair
- $x(t)$ réel impair ⇔ X(f) imaginaire impair
- $x(t)*y(t) \Leftrightarrow X(f).Y(f)$ et $x(t).y(t) \Leftrightarrow X(f)*Y(f)$!

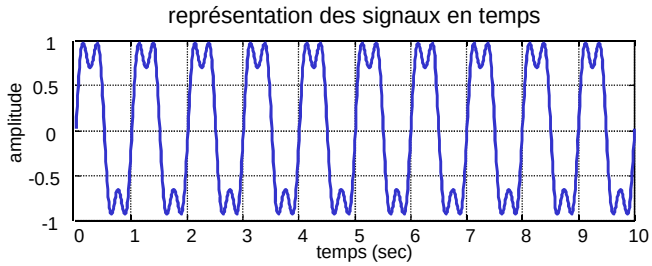
- $x(t)*\delta(t-t_0)= x(t-t_0) \Leftrightarrow X(f)\ exp(-2j\pi f\ t_0)$
- $x(t)\ exp(2\ j\ \pi\ t\ f_0) \Leftrightarrow X(f-f_0)$
- $x^*(t) \Leftrightarrow X^*(-f)$
- $x(at) \Leftrightarrow |a|^{-1}\ X(f/a)$
- $d^n x(t)/dt^n \Leftrightarrow (2\ j\ \pi f\ )^n\ X(f)$

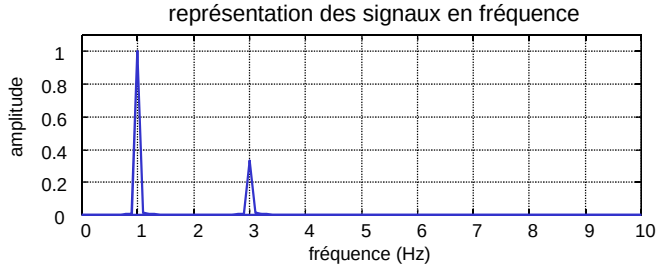
Quelques signaux et leur Transformée de Fourier

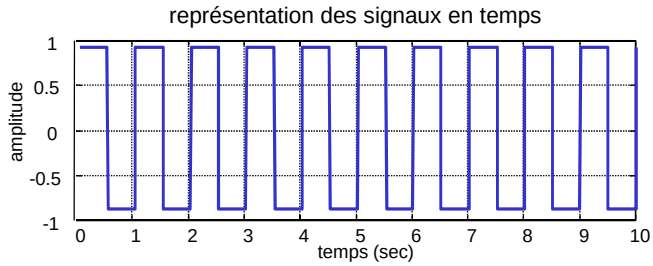
- $\delta(t) \Leftrightarrow 1$
- $1(t) \Leftrightarrow ½\ \delta(f) + 1/(2\ j\ \pi f\ )$
- $cos(2\pi f_0 t) \Leftrightarrow [\delta(f-f_0) + \delta(f+f_0)]/2$
- $sin(2\pi f_0 t) \Leftrightarrow [\delta(f-f_0) - \delta(f+f_0)]/2j$
- $\Sigma\delta(t+nT) \Leftrightarrow F_e\ \Sigma\delta(f+kF_e)$ avec $F_e=1/T$
- $Rect(t) \Leftrightarrow 2a.Sinc(\pi fa)$

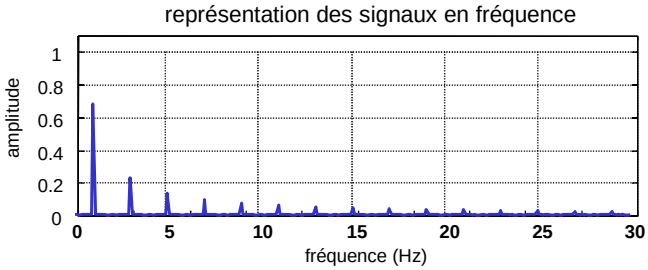

représentation des signaux en temps
amplitude
1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10
temps (sec)

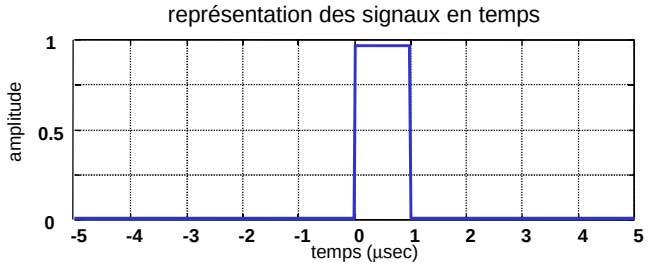

représentation des signaux en fréquence
amplitude
1
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10
fréquence (Hz)

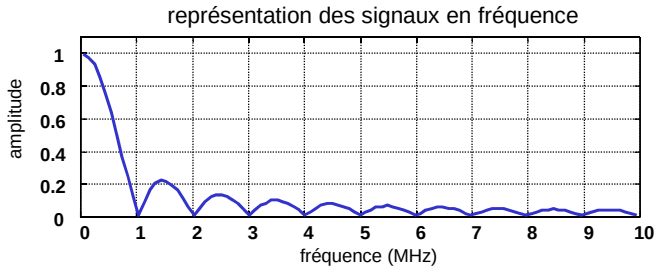

représentation des signaux en temps
amplitude
1
0.5
0
-0.5
-1
0 1 2 3 4 5 6 7 8 9 10
temps (sec)

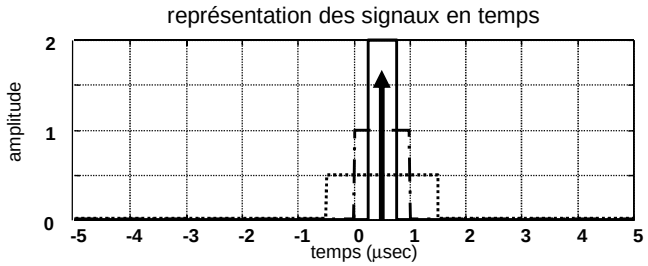

représentation des signaux en fréquence
amplitude
1
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10
fréquence (Hz)

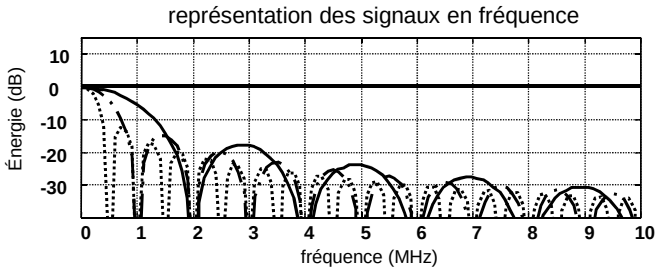

représentation des signaux en temps
amplitude
temps (sec)

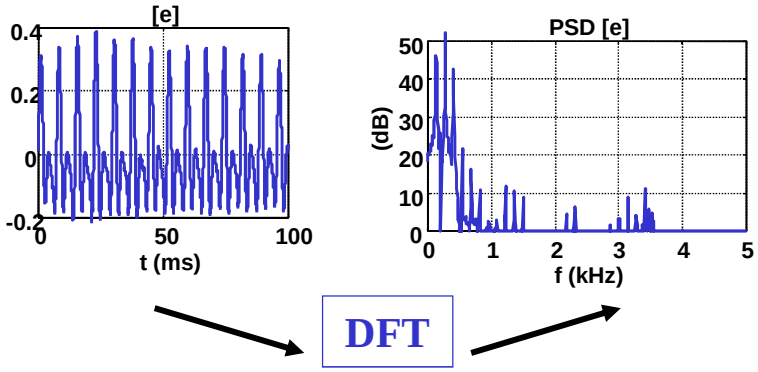

représentation des signaux en fréquence
amplitude
fréquence (Hz)


représentation des signaux en temps
amplitude
temps (µsec)


représentation des signaux en fréquence
amplitude
fréquence (MHz)





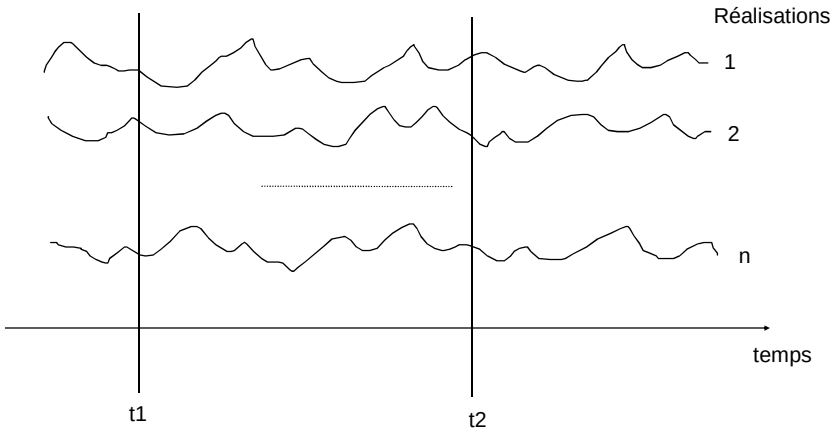


Que faire avec les signaux aléatoires ??

>> Théorie « signal aléatoire »

- Densité Spectrale de puissance / auto correlation
- Stationnarité / Ergodicité

>> Solutions numériques pratiques



Stationnaire >> stats(t1) = stats(t2)
Ergoqique >> propriétés temporelles = propriétés satistiques

Et l ’énergie d ’un signal ?

Identité de Parseval

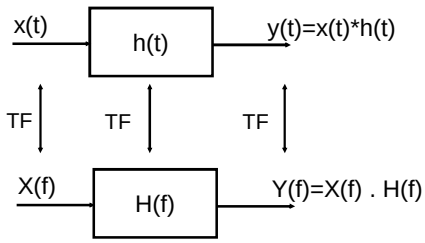
$$E = \int_{-\infty}^{+\infty} x^2(t).dt = \int_{-\infty}^{+\infty} |X(f)|^2.df = \frac{1}{2\pi}\int_{-\infty}^{+\infty} |X(w)|^2.dw$$

Densité Spectrale d ’Energie

Transformée de Fourier & Systèmes

Un SLTI va être caractérisé par sa réponse impulsionnelle h(t)

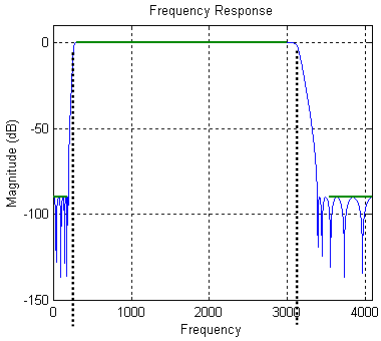
La transformée de Fourier de h(t) donne la réponse en fréquence du système H(f)

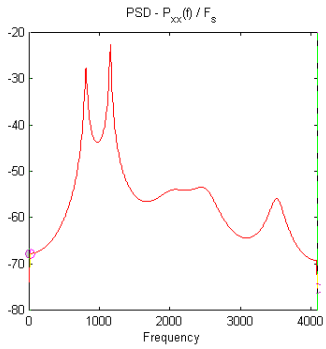


L 'inverse est aussi vrai

Bande passante et largeur de bande

- **Bande passante**
  - Caractérise un système
  - Module de la rép. en fréquence
  - Définie à -3dB (1/√2) (Pm/2)
- **Largeur de bande**
  - Caractérise un signal
  - Densité Spectrale
  - Espace des fréquences utiles !





Transformée de Fourier des signaux échantillonnés

Fréquence d 'échantillonnage $F_e=1/T_e$

La transformée de Fourier est discrète et donne un spectre **périodique**

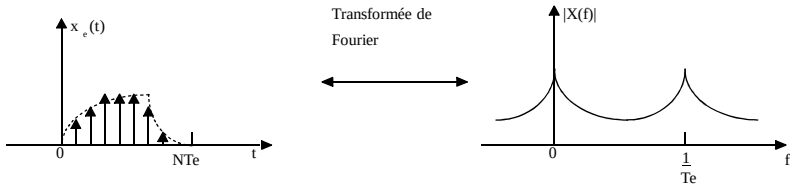
Le spectre est représenté de 0 à F_e ou de $-F_e/2$ à $F_e/2$

$$x_e(t) = \int_0^{F_e} X(f).\exp(j2\pi ft).df$$

$$X(f) = T_e.\sum_{t\in Z(T_e)} x_e(t)\exp(-j2\pi ft)$$

$$Z(T_e) = \{...,-3T_e,-2T_e,-T_e,0,T_e,2T_e,3T_e,...\}$$

Rque : pour des signaux discrets, on pose $T_e=1=F_e$, (et n=t)



Transformée de Fourier des signaux échantillonnés périodiques

Fréquence d 'échantillonnage $F_e=1/T_e$, Période du signal NT_e, Fréquence du signal $F=1/NT_e$

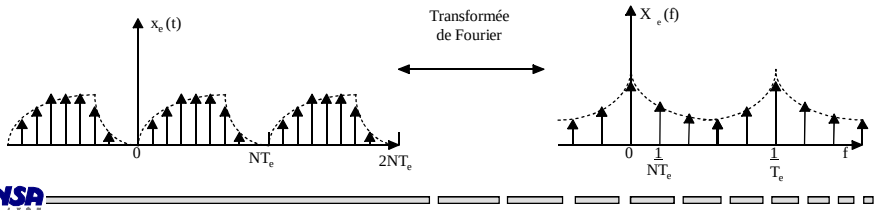
La transformée de Fourier est discrète et donne un spectre **périodique** et **discret**

Le spectre est constitué de N raies, il est représenté de 0 à F_e ou de $-F_e/2$ à $F_e/2$

$$x_e(t) = \frac{1}{NT_e}\sum_{f\in N(F)} X(f)\exp(j2\pi ft) \qquad N(F) = \{0,F,2F,3F,...NF\} \text{ avec } F = \frac{1}{NT_e}$$

$$X(f) = T_e.\sum_{t\in N(T_e)} x_e(t)\exp(-j2\pi ft) \qquad N(T_e) = \{0,T_e,2T_e,3T_e,...NTe\}$$

Rque : pour des signaux discrets, on pose $T_e=1$, (et n=t)



Du continu au discret, on échantillonne

• Échantillonnage idéal...

$$x_e(t) = x(t)\,\delta_T(t) = x(t)\sum_{k=-\infty}^{+\infty}\delta(t-kT) = \sum_{k=-\infty}^{+\infty}x[kT]\delta(t-kT)$$

• ...Transformée de Fourier...

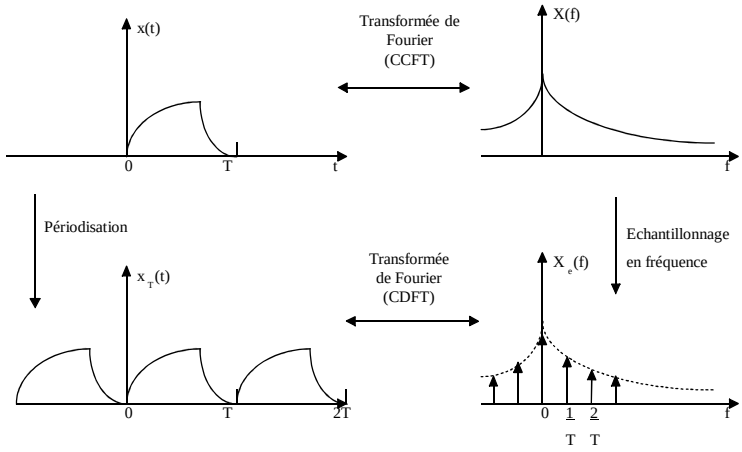
$$X_e(f) = \frac{1}{T}X(f) * \delta_{\frac{1}{T}}(f) = \frac{1}{T}\sum_{k=-\infty}^{+\infty}X(f-\frac{k}{T})$$

↖ *... périodisation en fréquence*.

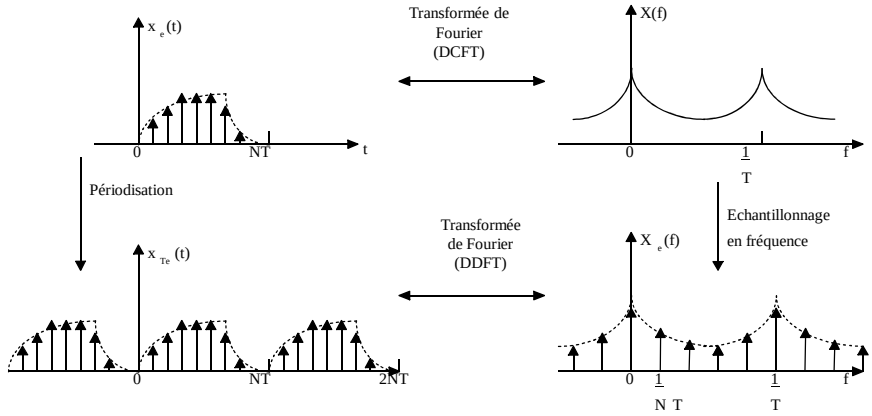
Échantillonnage temporel <=> Périodisation en fréquence

Échantillonnage en fréquence <=> Périodisation temporelle

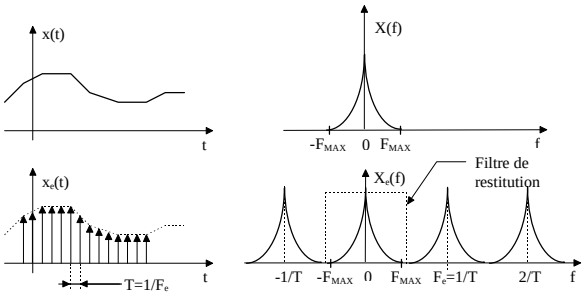
Tourne ...



et retourne ...



Echantillonnage : Théorème de Shannon



Si $\mathbf{F_e > 2\ F_{max}}$ alors les spectres périodisés ne se recouvrent pas

>> Reconstitution du signal analogique de départ théoriquement possible

Conséquence : la fréquence max que peut porter un signal échantilloné = Fe / 2

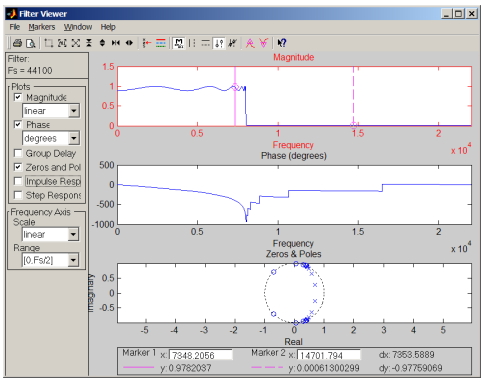
Et le numérique ?

L 'avant ? L 'après ? T_e ? Fréquence normalisée

Un signal numérique est fini (N points), pour faire sa TF, on le périodise implicitement

On rajoute éventuellement des 0 (Nz), pour avoir une TF sur (N+Nz) points.

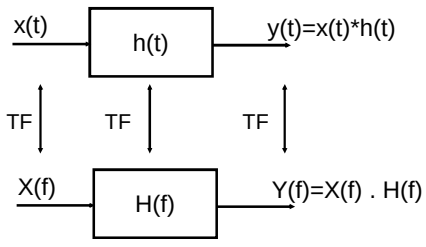
Des outils puissants : Matlab, Scilab, Labview …



Transformée de Fourier & Systèmes

Un SLTI va être caractérisé par sa réponse impulsionnelle h(t) : sortie du système quand l'entrée est une impulsion

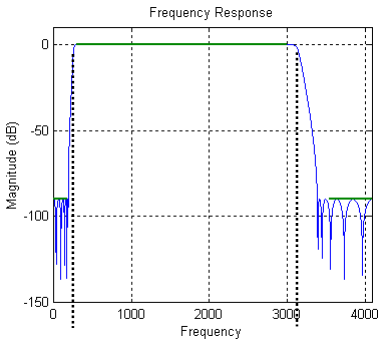
La transformée de Fourier de h(t) donne la réponse en fréquence du système H(f)

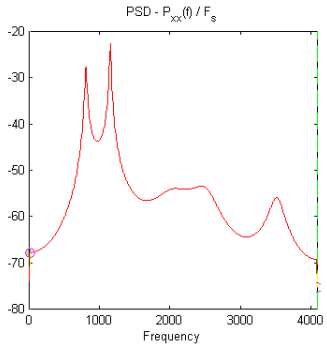


L 'inverse est aussi vrai

Bande passante et largeur de bande

- **Bande passante**
  - Caractérise un système
  - Module de la rép. en fréquence
  - Définie à -3dB (1/√2) (Pm/2)
- **Largeur de bande**
  - Caractérise un signal
  - Densité Spectrale
  - Espace des fréquences utiles !





Les différentes TF

Type of transform	Time axis	Frequency axis	Type of time-signal	Transform	Inverse Transform	Relation between parameters
DDFT	$\underline{N}$	$\underline{N}(F)$	finite-time[1] sequence	$\hat{x}(f) = \sum_{n \in \underline{N}} x(n)e^{-j2\pi fn}$	$x(n) = F \sum_{f \in \underline{N}(F)} \hat{x}(f)e^{j2\pi fn}$	$NF = 1$
DCFT	$\mathbb{Z}$	$[0, 1)$	infinite-time sequence	$\hat{x}(f) = \sum_{n \in \mathbb{Z}} x(n)e^{-j2\pi fn}$	$x(n) = \int_0^1 \hat{x}(f)e^{j2\pi fn}\, df$	
DDFT sampled	$\underline{N}(T)$	$\underline{N}(F)$	finite-time[2] sampled signal	$\hat{x}(f) = T \sum_{t \in \underline{N}(T)} x(t)e^{-j2\pi ft}$	$x(t) = F \sum_{f \in \underline{N}(F)} \hat{x}(f)e^{j2\pi ft}$	$NFT = 1$
DCFT sampled	$\mathbb{Z}(T)$	$[0, B)$	infinite-time sampled signal	$\hat{x}(f) = T \sum_{t \in \mathbb{Z}(T)} x(t)e^{-j2\pi ft}$	$x(t) = \int_{0-}^{B} \hat{x}(f)e^{j2\pi ft}\, df$	$BT = 1$
CDFT	$[0, P)$	$\mathbb{Z}(F)$	finite-time[3] continuous-time	$\hat{x}(f) = \int_{0-}^{P} x(t)e^{-j2\pi ft}\, dt$	$x(t) = F \sum_{f \in \mathbb{Z}(F)} \hat{x}(f)e^{j2\pi ft}$	$PF = 1$
CCFT	$\mathbb{R}$	$\mathbb{R}$	infinite-time continuous-time	$\hat{x}(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\, dt$	$x(t) = \int_{-\infty}^{\infty} \hat{x}(f)e^{j2\pi ft}\, df$	

1. Also applies to a single period of a periodic signal x defined on the time axis $\mathbb{Z}$ with period N.
2. Also applies to a single period of a periodic signal x defined on the time axis $\mathbb{Z}(T)$ with period NT.
3. Also applies to a single period of a periodic signal x defined on the time axis $\mathbb{R}$ with period P.

Propriétés

Property	Transform	Time signal	Transformed signal
Linearity	all	$\alpha x + \beta y$	$\alpha\hat{x} + \beta\hat{y}$
Convolution	DDFT, CDFT DCFT, CCFT	$x \odot y$ $x * y$	$\hat{x}\hat{y}$ $\hat{x}\hat{y}$
	DDFT, DCFT CDFT, CCFT	xy xy	$\hat{x} \odot \hat{y}$ $\hat{x} * \hat{y}$
Shift	all; $\theta \in \mathbb{T}$	$x(t+\theta), t \in \mathbb{T}$	$e^{j2\pi f\theta}\hat{x}(f), f \in \mathbb{F}$
	all; $\phi \in \mathbb{F}$	$e^{j2\pi\phi t}x(t), t \in \mathbb{T}$	$\hat{x}(f-\phi), f \in \mathbb{F}$
Periodicity	DDFT (sampled) CDFT DCFT	period NT period P not periodic	period NF not periodic period B
Differentiation	CDFT, CCFT DCFT, CCFT	$(Dx)(t), t \in \mathbb{T}$ $(-j2\pi t)x(t), t \in \mathbb{T}$	$(j2\pi f)\hat{x}(f), f \in \mathbb{F}$ $D\hat{x}(f), f \in \mathbb{F}$
Symmetry	all	real conjugate symmetric even odd real and even real and odd	conjugate symmetric real even odd real and even imaginary and odd

Notes:
(1) $\mathbb{T}$ is the time axis and $\mathbb{F}$ the corresponding frequency axis.
(2) For finite time and frequency axes the addition and subtraction in the shift property are modulo the length of the time or frequency axis.

CCFT connues

Time signal $x(t), t \in \mathbb{R}$	CCFT $\hat{x}(f), f \in \mathbb{R}$	Conditions
$x(t)$ $\int_{-\infty}^{\infty}\hat{x}(f)e^{j2\pi ft}\,df$	$\int_{-\infty}^{\infty}x(t)e^{-j2\pi ft}\,dt$ $\hat{x}(f)$	
$e^{-at}\mathbb{1}(t)$	$\frac{1}{j2\pi f + a}$	$a \in \mathbb{C}$, Re $(a) > 0$
$-e^{-at}\mathbb{1}(-t)$	$\frac{1}{j2\pi f + a}$	$a \in \mathbb{C}$, Re $(a) < 0$
$\frac{t^{k-1}e^{-at}}{(k-1)!}\mathbb{1}(t)$	$\frac{1}{(j2\pi f + a)^k}$	$k \in \mathbb{N}, a \in \mathbb{C}$, Re $(a) > 0$
$-\frac{t^{k-1}e^{-at}}{(k-1)!}\mathbb{1}(-t)$	$\frac{1}{(j2\pi f + a)^k}$	$k \in \mathbb{N}, a \in \mathbb{C}$, Re $(a) < 0$
$e^{-a\lvert t\rvert}$	$\frac{2a}{4\pi^2f^2 + a^2}$	$a \in \mathbb{C}$, Re $(a) > 0$
$\frac{1}{a^2 + t^2}$	$\frac{\pi}{a}e^{-2\pi a\lvert f\rvert}$	$a \in \mathbb{C}$, Re $(a) > 0$
e^{-at^2}	$\sqrt{\frac{\pi}{a}}e^{-\pi^2f^2/a}$	$a \in \mathbb{R}, a > 0$
$e^{-at}\cos(\omega t)\mathbb{1}(t)$	$\frac{j2\pi f + a}{(j2\pi f + a)^2 + \omega^2}$	$a, \omega \in \mathbb{R}, a > 0$
$e^{-at}\sin(\omega t)\mathbb{1}(t)$	$\frac{\omega}{(j2\pi f + a)^2 + \omega^2}$	$a, \omega \in \mathbb{R}, a > 0$
$\begin{cases}1, & -a \le t < a\\ 0, & \text{otherwise}\end{cases}$	$2a\ \mathrm{sinc}\ (2\pi fa)$	$a \in \mathbb{R}, a \ge 0$
$2a\ \mathrm{sinc}\ (2\pi at)$	$\begin{cases}1, & -a \le f < a\\ 0, & \text{otherwise}\end{cases}$	$a \in \mathbb{R}, a \ge 0$
$\begin{cases}1 - \lvert t\rvert/a, & -a \le t < a\\ 0, & \text{otherwise}\end{cases}$	$a\ \mathrm{sinc}^2\ (\pi fa)$	$a \in \mathbb{R}, a \ge 0$

CCFT généralisées connues

Time signal $x(t),\ t \in \mathbb{R}$	CCFT $\hat{x}(f),\ f \in \mathbb{R}$	Conditions
$\delta(t)$	1	
$\delta^{(k)}(t)$	$(j2\pi f)^k$	$k \in \mathbb{N}$
1	$\delta(f)$	
$(-j2\pi t)^k$	$\delta^{(k)}(f)$	$k \in \mathbb{N}$
$\delta(t+\theta)$	$e^{j2\pi f\theta}$	$\theta \in \mathbb{R}$
$e^{j2\pi f_o t}$	$\delta(f-f_o)$	$f_o \in \mathbb{R}$
$\sin(2\pi f_o t)$	$-\frac{j}{2}[\delta(f-f_o)-\delta(f+f_o)]$	$f_o \in \mathbb{R}$
$\cos(2\pi f_o t)$	$\frac{1}{2}[\delta(f-f_o)+\delta(f+f_o)]$	$f_o \in \mathbb{R}$
$\mathcal{U}(t)$	$\frac{1}{2}\delta(f)+\frac{1}{j2\pi f}$	
$\operatorname{sign}(t)$	$\frac{1}{j\pi f}$	
$t\,\mathcal{U}(t)$	$\frac{j}{4\pi}\delta^{(1)}(f)-\frac{1}{4\pi^2 f^2}$	
$\lvert t\rvert$	$\frac{-1}{2\pi^2 f^2}$	
$\sum_{n=-\infty}^{\infty} a_n \delta(t-nT)$	$\sum_{k=-\infty}^{\infty} a_k e^{-j2\pi kfT}$	$\{a_k\}$ of polynomial growth, $T \in \mathbb{R}, T \neq 0$
$\sum_{n=-\infty}^{\infty} a_n e^{j2\pi nf_o t}$	$\sum_{k=-\infty}^{\infty} a_k \delta(f-kf_o)$	$\{a_k\}$ of polynomial growth, $f_o \in \mathbb{R}, f_o \neq 0$
$\sum_{n=-\infty}^{\infty} \delta(t+nT)$	$\frac{1}{T}\sum_{k=-\infty}^{\infty} \delta\left(f+\frac{k}{T}\right)$	$T \in \mathbb{R}, T \neq 0$
$\sum_{n=-\infty}^{\infty} e^{j2\pi nf_o t}$	$f_o \sum_{k=-\infty}^{\infty} e^{j2\pi kf/f_o}$	$f_o \in \mathbb{R}, f_o \neq 0$

DCFT connues

Time signal $x(n),\ n \in \mathbb{Z}$	DCFT $\hat{x}(f),\ f \in [0^-, 1)$ or $f \in \mathbb{R}$	Conditions
$x(n)$	$\sum_{n=-\infty}^{\infty} x(n)e^{-j2\pi fn}$	
$\int_{0^-}^{1} \hat{x}(f)e^{j2\pi fn}\,df$	$\hat{x}(f)$	
$\begin{cases}1, & n=0\\ 0, & \text{otherwise}\end{cases}$	1	
$\begin{cases}1, & n=0,1,\cdots,M-1\\ 0, & \text{otherwise}\end{cases}$	$e^{-j\pi f(M-1)}d_M(f)$	$M \in \mathbb{N}$
1	$\sum_{k=-\infty}^{\infty} \delta(f-k)$	
$\begin{cases}a^n, & n=0,1,\cdots,M-1\\ 0, & \text{otherwise}\end{cases}$	$\frac{1-a^M e^{-j2\pi fM}}{1-ae^{-j2\pi f}}$	$a \in \mathbb{C}, M \in \mathbb{N}$
$a^n\mathcal{U}(n)$	$\frac{1}{1-ae^{-j2\pi f}}$	$a \in \mathbb{C}, \lvert a\rvert < 1$
$-a^n\mathcal{U}(-n-1)$	$\frac{1}{1-ae^{-j2\pi f}}$	$a \in \mathbb{C}, \lvert a\rvert > 1$
$\begin{cases}e^{j2\pi f_o n}, & n=0,1,\cdots,M-1\\ 0, & \text{otherwise}\end{cases}$	$e^{-j\pi(f-f_o)(M-1)}\,d_M(f-f_o)$	$M \in \mathbb{N}, f_o \in \mathbb{R}$
$e^{j2\pi f_o n}$	$\sum_{k=-\infty}^{\infty} \delta(f-f_o-k)$	$f_o \in \mathbb{R}$

Notes:

(1) Dirichlet's kernel d_M is defined as $d_M(\xi) = M\frac{\operatorname{sinc}(\pi\xi/M)}{\operatorname{sinc}(\pi\xi)}$, $\xi \in \mathbb{R}$ (see 7.3.11).

(2) If the sequence x defined on $\mathbb{Z}$ has the DCFT $\hat{x}$ defined on $[0, 1)$, the sampled signal x^* defined on $\mathbb{Z}(T)$ by

$$x^*(nT) = x(n), \qquad n \in \mathbb{Z},$$

has the sampled DCFT $\hat{x}^*$ defined on $[0, 1/T)$ by

$$\hat{x}^*(f) = T\,\hat{x}(fT), \qquad f \in [0, 1/T).$$

CDFT connues

Name[1]	Time signal $x(t),\ t \in [0,\ 1)$	CDFT, Fourier coefficients $\hat{x}(f),\ f \in \mathbb{Z}$
	$x(t)$ $\sum_{f \in \mathbb{Z}} \hat{x}(f)e^{j2\pi ft}$	$\int_{0^-}^{1} x(t)e^{-j2\pi ft}\,dt$ $\hat{x}(f)$
Rectangular pulse $a \in [0,\ 1]$	$\begin{cases} 1, & 0 \le t < a \\ 0, & a \le t < 1 \end{cases}$	$\begin{cases} a, & f = 0 \\ ae^{-j\pi fa}\ \mathrm{sinc}\ (\pi fa), & \text{otherwise} \end{cases}$
Square wave	$\begin{cases} 1, & 0 \le t < \frac{1}{2} \\ 0, & \frac{1}{2} \le t < 1 \end{cases}$	$\begin{cases} \frac{1}{2}, & f = 0 \\ 0, & f \text{ even},\ f \neq 0 \\ -j/f\pi, & f \text{ odd} \end{cases}$
Triangular pulse	$\begin{cases} t, & 0 \le t < \frac{1}{2} \\ 1-t, & \frac{1}{2} \le t < 1 \end{cases}$	$\begin{cases} \frac{1}{4}, & f = 0 \\ 0, & f \text{ even},\ f \neq 0 \\ -1/\pi^2 f^2, & f \text{ odd} \end{cases}$
Sawtooth	$t,\ 0 \le t < 1$	$\begin{cases} \frac{1}{2}, & f = 0, \\ -1/j2\pi f, & \text{otherwise} \end{cases}$
Half-wave rectified sine	$\begin{cases} \sin(2\pi t), & 0 \le t < \frac{1}{2} \\ 0, & \frac{1}{2} \le t < 1 \end{cases}$	$\begin{cases} \pm\frac{1}{4}j, & f = \pm 1 \\ 0, & f \text{ odd},\ f \neq \pm 1 \\ -2/\pi(f^2-1), & f \text{ even} \end{cases}$
Full-wave rectified sine	$\lvert\sin(2\pi t)\rvert$	$\begin{cases} 0, & f \text{ odd} \\ -4/\pi(f^2-1), & f \text{ even} \end{cases}$
Infinite comb	$\sum_{k=-\infty}^{\infty} \delta(t-k)$	1

1. See Fig. 7.5 for plots of the various signals.

Note: If x is defined on $[0,\ 1)$ with CDFT $\hat{x}$, the CDFT $\hat{x}_P$ of the signal x_P defined on $[0,\ P)$ by

$$x_P(t) = x(t/P), \qquad t \in [0,\ P),$$

is given by

$$\hat{x}_P(f) = P\hat{x}(fP), \qquad f \in \mathbb{Z}(1/P).$$

DDFT connues

Time signal $x(n),\ n \in \underline{N}$	DDFT, Fourier coefficients $\hat{x}(f),\ f \in \underline{N}(1/N)$	Conditions
$x(n)$ $\sum_{f \in \underline{N}(1/N)} \hat{x}(f)e^{j2\pi fn}$	$\sum_{n=0}^{N-1} x(n)e^{-j2\pi fn}$ $\hat{x}(f)$	
$\begin{cases} 1, & n = 0 \\ 0, & \text{otherwise} \end{cases}$ $\begin{cases} 1, & n = 0,\ 1,\ \cdots,\ M-1 \\ 0, & \text{otherwise} \end{cases}$ 1	1 $e^{-j\pi f(M-1)}d_M(f)$ $\begin{cases} N, & f = 0 \\ 0, & \text{otherwise} \end{cases}$	$M \in \underline{N}$
$\begin{cases} a^n, & n = 0,\ 1,\ \cdots,\ M-1 \\ 0, & \text{otherwise} \end{cases}$ $\begin{cases} e^{j2\pi f_o n}, & n = 0,\ 1,\ \cdots,\ M-1 \\ 0, & \text{otherwise} \end{cases}$	$\frac{1 - a^M e^{-j2\pi fM}}{1 - ae^{j2\pi f}}$ $e^{-j\pi(f-f_o)(M-1)}d_M(f - f_o)$	$M \in \underline{N},\ a \in \mathbb{C}$ $M \in \underline{N},\ f_o \in \mathbb{R}$
$e^{j2\pi f_o n}$	$\begin{cases} N, & f = f_o \\ 0, & \text{otherwise} \end{cases}$	$f_o \in \underline{N}(1/N)$

Notes:

(1) Dirichlet's kernel d_M is defined as $d_M(\xi) = M\frac{\mathrm{sinc}\ (\pi\xi/M)}{\mathrm{sinc}\ (\pi\xi)}$, $\xi \in \mathbb{R}$ (see 7.3.11).

(2) If the sequence x defined on $\underline{N}$ has the DDFT $\hat{x}(f),\ f \in \underline{N}(1/N)$, the sampled signal x^* defined on $\underline{N}(T)$ by

$$x^*(nT) = x(n), \qquad n \in \underline{N},$$

has the sampled DDFT $\hat{x}^*$ defined on $\underline{N}(1/NT)$ by

$$\hat{x}^*(f) = T\hat{x}(fT), \qquad f \in \underline{N}(1/NT).$$

Séance 7 – TP
Découverte de MATLAB

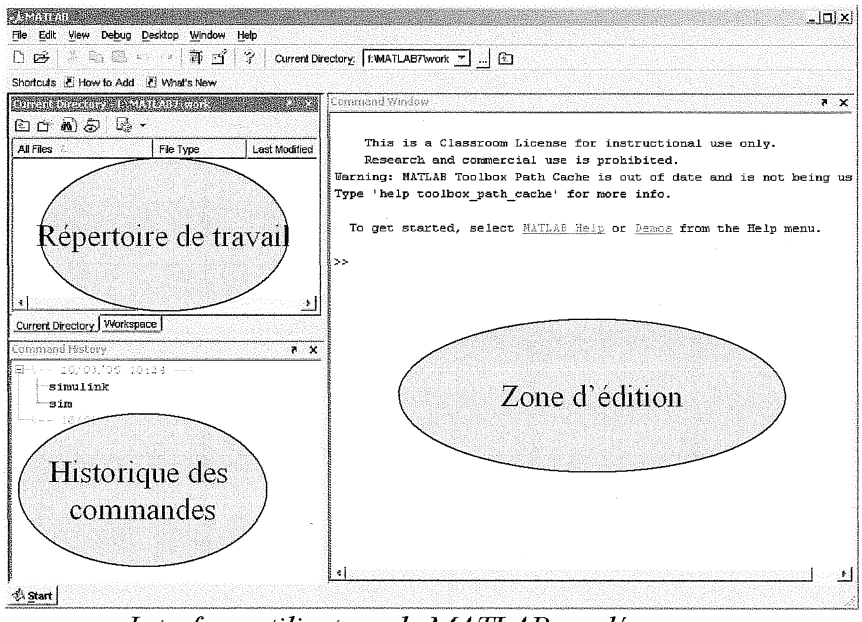
1. Prise en main de Matlab

MATLAB est un logiciel de calcul matriciel qui propose une syntaxe simple.

Avec ses nombreuses bibliothèques de fonctions spécialisées (*toolboxes*), MATLAB peut être aussi considéré comme un langage de programmation adapté pour de très nombreux problèmes scientifiques (simulation de systèmes physiques, contrôle de robots, calcul statistique, ...) dont le traitement du signal.

MATLAB est un « *interpréteur* »: les instructions que vous lui tapez sont interprétées et exécutées ligne par ligne ; pas besoin de compilateur !
Il fonctionne selon deux modes:

- **mode interactif:** MATLAB exécute les instructions au fur et à mesure que vous les tapez.
- **mode exécutif:** MATLAB exécute ligne par ligne un *"fichier M"*, c'est-à-dire, un programme en langage MATLAB.



Interface utilisateur de MATLAB au démarrage

1.1. Opérations de base

Par défaut, MATLAB réagit comme une calculatrice : à l'invite de commande « >> », tapez « `1+1` », « `1e5 / 10` », « `2^3` », ... et il vous affiche le résultat immédiatement sous la forme « `ans= ...` ». On peut faire plusieurs opérations à la suite en les séparant par des « `;` ». Attention, les instructions suivies des « `;` » n'affichent pas leurs résultats.

En fait, il stocke automatiquement le résultat du dernier calcul demandé dans la variable « `ans` ». Tapez « `ans` » et il vous réaffiche le résultat. Vous pouvez stocker le résultat d'une

opération dans une variable de votre choix, par ex : « `ageToto = 3*5` ». Cette variable reste mémorisée jusqu'à la fermeture de MATLAB.

A tout moment, vous pouvez connaître le nom et le type des variables en mémoire avec la commande : « **whos** » et effectuer des opérations entre variables : « `agePapaToto=2*ageToto` »

Les nombres manipulés sont des réels, voire des nombres complexes : « `1+i` ».

La plupart des fonctions mathématiques classiques sont présentes : « `cos` », « `sin` », « `exp` », « `log` », « `tan` », « `cosh` », « `conj` », « `abs` » , « `angle` », ...

1.2. Vecteurs

Pour représenter un signal, il est nécessaire d'utiliser des vecteurs : des variables de dimension > 1×1 : Par exemple, tapez « `t= 1 :10` » et visualisez le résultat.

Tapez « `t= 1 :2 :10` », qu'obtenez-vous ?

Vous pouvez ensuite récupérer une valeur particulière d'un vecteur : « `t(3)` » par exemple et même la modifier : « `t(3)=10` ». Attention, le premier indice est **1** (et non 0 comme dans la plupart des langages de programmation).

Si vous voulez récupérer une partie du vecteur : « `t(1 :3)` » ; de même vous pouvez modifier cette partie : « `t(1:3)=[8,9,10]` » ou encore : « `t(1:3)=t(1:3)+1` » et aussi : « `t(4:6)=t(1:3)*3` »

On ne peut additionner que 2 vecteurs strictement de la même taille. Pour obtenir la longueur d'un vecteur : « `length(t)` »

1.3. Courbes

Une courbe est l'association entre 2 vecteurs de même longueur. L'instruction « `plot(x,y)` » affiche la courbe représentée par x en abscisse et y en ordonnées. Par exemple : « `x=0:100;y=sin(2*pi*x/100);plot(x,y)` ».

Vous pouvez superposer plusieurs courbes : « `plot(x1,y1, x2, y2, ...)` ». Superposez un sinus et un cosinus.

Vous pouvez modifier le style de chacune des courbes : « `plot(x1,y1, 'b.', x2, y2,'co' ...)` ».

Pour connaître le codage du style des courbes : « `help plot` » comme pour tout autre fonction dont vous ne savez pas vous servir... « `help` » tout court vous affiche le sommaire de l'aide complète de MATLAB... Vous verrez, il y a de quoi s'occuper pendant pas mal de TP...

Pour effectuer un zoom manuel sur la courbe en utilisant les outils en haut de la fenêtre mais vous pouvez également choisir de n'afficher qu'une portion de la courbe en extrayant que les valeurs dans un intervalle de temps choisi : « `inter=5:10;plot(t(inter),y(inter))` »

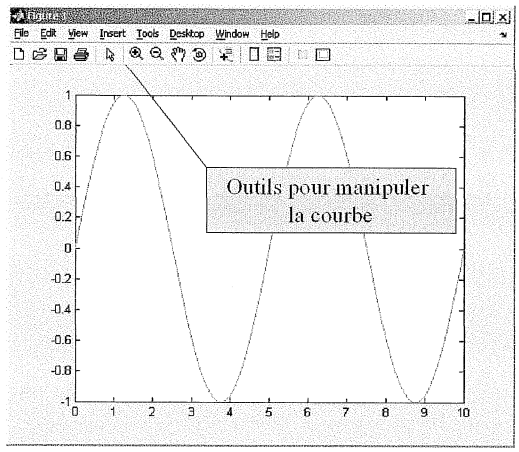


Figure représentant l'évolution d'un signal ici sinusoïdal

2. Synthèse et affichage d'un signal en temps

2.1. résolution temporelle,

Pour synthétiser un signal, il vous faut une « *base de temps* ». Dans la réalité et dans les formules mathématiques, le temps est continu mais MATLAB ne sait pas représenter un signal continu, uniquement des vecteurs. On va donc utiliser un vecteur dont les valeurs représentent l'évolution du temps à une période de scrutation donnée. Par exemple, le vecteur « `t=0 :10` » représente le temps de 0 à 10 secondes, par intervalle d'une seconde.

Tapez « `t=0:10;y=sin(2*pi*t/10);plot(t,y,'o')` »
Que constatez-vous ?

Ici la variable « y » représente un signal sinusoïdal échantillonné avec quelle période d'échantillonnage ?

Diminuez la période d'échantillonnage jusqu'à obtenir un signal qui ressemble à un sinus.

2.2. fréquence du signal

Quelle est la fréquence du signal sinusoïdal précédent ?

Comment augmenter ou diminuer sa fréquence ?

Que constatez-vous quand vous augmentez la fréquence du signal pour une même période d'échantillonnage ?

2.3. fréquence d'échantillonnage et fréquence réduite

La fréquence d'échantillonnage, f_e, c'est l'inverse de la période d'échantillonnage.

Pour pouvoir représenter un signal proprement avec MATLAB, il faut prendre soin de choisir une fréquence d'échantillonnage grande devant la fréquence des signaux manipulés.

Le rapport fréquence du signal sur fréquence d'échantillonnage s'appelle fréquence réduite ou fréquence normalisée.

2.4. Signal issu de la vie courante

Très souvent, MATLAB est utilisé pour analyser des signaux récupérés par des capteurs puis numérisés. Dans notre cas, nous allons nous amuser avec un son :

Placez le répertoire courant de MATLAB (fenêtre en haut à gauche dans le répertoire C:\WINNT\Media). Tapez ensuite : « `[y,fs,nbits]=wavread('chimes.wav');` »

`y` contient, sous forme d'une matrice 2 colonnes, 13921 lignes, les échantillons de l'amplitude du son stéréo pour les deux canaux (1 par colonne), `fs` est la fréquence d'échantillonnage du signal sonore et `nbits` le nombre de bits utilisés pour coder son amplitude.

Pour représenter le signal, il faut recréer le vecteur temps. Comment faire ?

Extrayez les deux canaux : « `yg=y(:,1);yd=y(:,2);` » puis affichez les ensemble.

Le signal est-il bien stéréo ? Faîtes un zoom dessus, à quoi ressemble-t'il de plus près ?

3. Synthèse et affichage d'un signal en fréquence

Tout signal, périodique ou non, peut se représenter sous la forme d'une somme infinie de signaux élémentaires sinusoïdaux. C'est une représentation duale de la représentation temporelle illustrée jusqu'ici dans ce TP, en d'autres mots, un point de vue différent sur le même signal.

En théorie, la Transformée de Fourier permet de passer de la représentation temporelle à la représentation fréquentielle. Avec MATLAB, nous utiliserons la FFT : *Fast Fourier Transform*, qui est une adaptation logicielle de la transformée de Fourier pour un signal échantillonné à la fois dans le temps et à la fois dans l'échelle de fréquence.

La fonction « `fft(signal, nbrePointsFrequence)` » effectue ce calcul. Il faut simplement lui fournir le nombre de points à utiliser pour représenter l'ensemble des fréquences de 0 à f_e, (généralement une puissance de 2, **1024** par exemple, pour des questions d'optimisation du calcul).

Application : écrivez un script (un ensemble de commandes rassemblées dans un fichier) pour illustrer cette représentation :

- Créer un signal sinusoïdal échantillonné à 1000Hz et de fréquence 25Hz sur une durée de 10s.
- Afficher le module de la FFT de ce signal. Que constatez-vous ?
- La partie du spectre qui nous intéresse réside entre les fréquences 0 et $f_e/2$ (le reste est redondant). Créez donc un vecteur fréquence correspondant et affichez le module de la FFT en fonction de ce vecteur.

3.1. résolution fréquentielle.

Utilisez un signal avec deux fréquences très proches : 25Hz et 30Hz et faîtes varier `nbrePointsFrequence`. Quelle est l'influence de `nbrePointsFrequence` sur la représentation fréquentielle ?

4. Bibliographie :

- Les signaux audio numériques et la FFT : http://www.unilim.fr/pages_perso/jean.debord/math/fourier/fft.htm#section2
- La FFT : http://cermics.enpc.fr/polys/oap/node86.html
- MATLAB: http://www.gel.ulaval.ca/~lehuy/intromatlab/

MATLAB

Les commandes indispensables

`help`	donne toute la liste des aides
`help sujet`	affiche l'aide sur le *sujet*
`commande ;`	le ; interdit l'affichage du résultat de la commande à l'écran
`quit, exit`	arête le programme

Commandes sur fichiers et répertoires

`cd nomrépertoire`	changer de répertoire de travail (cd .. : remonter d'un répertoire)
`pwd`	afficher le nom du répertoire courant
`dir, ls`	afficher le contenu du répertoire courant
`delete nomfichier`	supprimer le fichier *nomfichier*
`load nomfichier`	importe les variables du fichier *nomfichier.mat (format matlab)*
`load nomfichier.dat`	importe la matrice *nomfichier* du fichier *nomfichier.dat* (format texte)
`load nomfichier X Y`	importe uniquement les variables *X* et *Y* du fichier *nomfichier.mat*
`nomfichier`	exécute le scrip contenu dans le fichier *nomfichier.m* (commandes, définition variables)
`save nomfichier`	sauve toutes les variables dans le fichier *nomfichier.mat*
`save nom fichier X Y`	sauve uniquement les variables *X* et *Y* dans le fichier *nomfichier.mat*
`'texte quelconque'`	les ' ' sont utilisés pour indiquer que le texte quelconque doit être considéré comme un seul élément. Ex : cd 'mes documents' ou x = 'load fichier.m' (x est une matrice)

Gérer les variables

`who`	liste les variables utilisées
`whos`	donne la liste des variables ainsi que leur dimension
`clear`	supprime toutes les variables de l'espace de travail
`clear X Y`	supprime seulement les variables *X* et *Y*
`= =, <, >, <=, >=`	opérateurs de comparaison de variables (aide : *help eq, lt, gt, le, ge*)
`x = y`	affectation : x prend la valeur de y (aide : *help punct*)
`a:b`	intervalle [a, *a+1* ... *b*] (aide : *help colon*)
`x=y(:,k)`	le vecteur x est égal toute la $k^{ième}$ colonne de la matrice *y*
`x=y(a:b,k)`	le vecteur *x* est égal à la $k^{ième}$ colonne de la matrice y, mais uniquement du composant numéro *a* au composant numéro *b*
`max(X)`	donne le plus grand élément du vecteur *X*
`max(y(a:b,k))`	donne le plus grand élément de la $k^{ième}$ colonne de la matrice y, mais uniquement du composant numéro *a* au composant numéro *b*
`min(X)`	si *X* est un vecteur, max(*X*) donne le plus petit élément de ce vecteur

Fonctions graphiques

`plot(X,Y,'cs')`	dessine le graphe de *Y* en fonction de *X* suivant la couleur *c* et en utilisant le symbole *s*. *c* peut prendre les valeurs : y (jaune), m (magenta), c (cyan), r (rouge), g (vert), b (bleu), w blanc) et b (noir). s peut être entre autres : . (point), - (tiret), -. (tiret-point)
`title('texte')`	ajoute le *texte* comme titre de la figure
`xlabel('texte')`	ajoute le *texte* comme légende de l'axe des abcisses
`ylabel('texte')`	ajoute le *texte* comme légende de l'axe des ordonnées
`gtext('texte')`	placera le *texte* à partir du point sélectionné à la souris
`legend(texte1, texte2…)`	place une légende sur la figure courante. texte1, texte2, etc sont sur des lignes différentes
`legend(…,pos)`	place la légende à la position *pos*. *pos* peut prendre la valeur : 0 : placement automatic « le meilleur » (conflit minimum avec les données) 1 : dans le coin en haut à droite - 2 : dans le coin en haut à gauche 3 : dans le coin en bas à gauche - 4 : dans le coin en bas à droite -1 : à la droite de la figure
`axis([`X_{min} X_{max} Y_{min} Y_{max}`])`	retrace la figure dans les limites imposées sur chaque axe
`zoom`	permet de zoomer sur une zone particulière de la figure. La zone est sélectionnée par le bouton gauche de la souris, et désactivée par le bouton droit.
`grid on`	ajoute des lignes sur la figure
`figure`	crée une nouvelle figure
`figure(numéro)`	la figure portant le numéro *num* devient la figure courante (celle ou seront dessinées les courbes). La figure est créée si elle n'existait pas.
`hold on`	autorise le dessin des prochaines fonctions sur la figure sans effacer les précédentes
`close`	ferme la figure courante
`close(num)`	ferme la figure portant le numéro *num*
`close all`	ferme toutes les figures

Programmation

`for i=min:max … end`	répétition en boucle. Pour *i* = *min* à *i* = *max* faire …
`Break`	termine immédiatement le traitement d'une boucle for
`if cond1 … elseif cond2 … else … end`	exécution conditionnelle. Si condition *cond1* faire … sinon si *cond2* faire …sinon faire…
`% texte`	commentaires

OT Bases du signal

Séance machine n°3

Résumé
On propose l'analyse en temps et en fréquence d'un signal composé de 2 impulsions gaussiennes de fréquence centrale et de bande passante différentes. Après avoir mesurer les caractéristiques en temps et en fréquence des signaux, la notion de filtrage sera introduite dans le but d'identifier les fréquences appartenant à chacune des impulsions gaussiennes.

Canevas de la séance

1 Analyse d'impulsions gaussiennes en temps et en fréquence

- Générer une impulsion gaussienne d'équation $x(n) = Ae^{-\pi B^2(n-n_0)^2}$, $n \in [0, N-1]$, et calculer son spectre $X(f) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi fn}$, $f \in [0,1[$, avec

  A l'amplitude, 1/B la demie largeur de l'impulsion à mi-hauteur, n_0 le centre de l'impulsion, N le nombre de points

  Choisir les paramètres suivants puis les modifier pour visualiser les effets sur le signal et son spectre.
  A1 = 0.5; B1 = 0.05; n1 = 40;

- Générer une impulsion gaussienne modulée par une porteuse sinusoïdale. L'équation est : $x(n) = Ae^{-\pi B^2(n-n_0)^2}\cos(2\pi f_0(n-n_0))$, $n \in [0, N-1]$, et calculer son spectre $X(f) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi fn}$, $f \in [0,1[$, avec f_0 la fréquence de la porteuse.

  Fixer la fréquence porteuse à $f_1 = 0.1$ puis modifier cette valeur pour visualiser les effets sur le signal et son spectre.

- Générer une somme de 2 impulsions gaussiennes modulées $x'(n)$ perturbées par un bruit blanc gaussien additif $b(n)$.
  $x'(n) = x_1(n) + x_2(n)$
  Les valeurs des paramètres du signal $x_1(n)$ sont celles du paragraphe précédent. Les valeurs pour $x_2(n)$ sont : A2 = 1; B2 = 0.1; n2 = 80; f2 = 0.2
  Le signal $x(n)$ est :

  $$x(n) = x'(n) + \sigma\, b(n) \quad \text{avec } \sigma^2 = \frac{10^{-RSB/10}}{N}\sum_{n=0}^{N-1} x'^2(n)$$

  Observer le signal $x(n)$ et son spectre pour différentes valeurs du paramètre RSB (Rapport Signal à Bruit) exprimé en décibel. On pourra faire varier ce paramètre entre -10 dB et +40 dB par exemple.

- Calculer le centroïde du spectre du signal $x(n)$ en estimant la fréquence moyenne à partir de la densité spectrale d'énergie du signal $P_{xx}(f) = |X(f)|^2$.

$$\bar{f} = \frac{\int_{-\infty}^{+\infty} f P_{xx}(f) df}{\int_{-\infty}^{+\infty} P_{xx}(f) df}$$

Pour obtenir la fréquence centrale de chaque impulsion gaussienne, il est nécessaire de séparer leur contenu spectral. Cette séparation s'obtient par filtrage.

2 Filtrage des impulsions gaussiennes

Toutes les étapes décrites ci-dessous seront à réaliser deux fois. Une première fois pour isoler l'impulsion basse fréquence, puis une seconde fois, pour isoler l'impulsion haute fréquence.

- Conception d'un filtre
  On propose d'utiliser la méthode de la fenêtre pour concevoir le filtre. Cette méthode est mise en œuvre dans la fonction *fir1* de Matlab. On choisira un nombre de coefficients de $N_c = 40$ et une fréquence de coupure $f_c = 0.15$ de façon à isoler l'impulsion basse fréquence.
  Observer la réponse impulsionnelle du filtre et son spectre.

- Application du filtre
  Le signal filtré $y(n)$ s'obtient en calculant l'opération de convolution entre le signal non filtré $x(n)$ et la réponse impulsionnelle du filtre $h(n)$. Ce calcul est mis en œuvre dans la fonction Matlab *filter* et s'exprime par l'équation : $y(n) = \sum_{i=0}^{N_c-1} h(i)x(n-i)$
  Observer le signal filtré et son spectre.

- Calcul du centroïde du spectre du signal $y(n)$

ANALYSE DE FOURIER A COURT TERME

Didier VRAY- CREATIS - INSA - Lyon

Le but du travail est d'analyser le signal contenu dans le fichier **tone1.mat.** L'objectif est de décoder ce signal qui est constitué d'une succession de sinusoïdes de fréquence constante sur une courte durée. Le signal inconnu résulte de l'encodage d'une séquence de digits représentant des fréquences différentes suivant la correspondance suivante :

Digit	0	1	2	3	4	5	6	7	8	9
F(Hz)	250	750	1250	1750	2250	2750	3250	3750	4250	4750

Le signal est échantillonné à la fréquence f = 10kHz.

1 - Lire le fichier avec la commande **Load** de Matlab.
Représenter le signal. Graduer correctement l'axe temporel.
Quel est le nombre d'échantillons du signal? Quelle est sa durée? Repérer les différentes composantes temporelles du signal. Evaluer visuellement la durée de ces composantes.

2 - Calculer la Transformée de Fourier du signal. Choisir le nombre de points de la transformée de Fourier. Représenter le spectre fréquentiel correspondant aux fréquences positives. Graduer correctement l'axe fréquentiel. Analyser le contenu fréquentiel. Combien de composantes spectrales peut-on mettre en évidence. A quelle fréquence correspond chacune d'entre elles.

3 - Calculer séparément la Transformée de Fourier de plusieurs tranches du signal. Montrer que chacune de ces tranches n'a pas le même contenu fréquentiel.

4 - Il s'agit maintenant de décoder la suite de digits codés en fréquence en prenant successivement des morceaux du signal. Proposer un programme qui permet le décodage de la séquence en calculant la Transformée de Fourier de chaque tranche et en détectant le pic de fréquence. Pour cela, le signal sera découpé "manuellement", les numéros des échantillons de début et de fin de chaque tranche seront fournis au programme suite à l'analyse temporelle précédente.

5 - Proposer une analyse temps-fréquence par le spectrogramme.

L'expression du spectrogramme est : $S_x(t,f;h)=\left|\int_{-\infty}^{+\infty} x(\tau)h^*(\tau-t)e^{-j2\pi f\tau}d\tau\right|^2$

x est le signal. h est la fenêtre d'analyse de durée limitée. Le résultat est présenté sous la forme d'une image avec l'axe temporel sur l'horizontal, l'axe fréquentiel sur la verticale. Chaque pixel est codé en couleur en fonction de l'amplitude de $S_x(t,f)$.
On pourra utiliser la fonction Specgram de Matlab qui calcule et affiche le spectrogramme du signal x :
specgram(x,Nfft,Fe,FenetreAnalyse, Nrecouvre)

Nfft est le nombre de points de la Transformée de Fourier
Fe est la fréquence d'échantillonnage
FenetreAnalyse est la fenêtre d'analyse de longueur N points
Nrecouvre est le nombre de points de recouvrement entre les fenêtres d'analyse (maximum N-1)

Retrouver le code de la séquence à partir de l'image temps-fréquence. Justifier le choix des paramètres.

6 - D'autres signaux fournis peuvent être analysés avec le spectrogramme :
Le fichier VOWELS.MAT contient les sons des voyelles AEIOU parlées successivement (en anglais).
B3PULSES.MAT contient des impulsions gaussiennes. Il s'agit de les localiser en temps et en fréquence.
Le fichier FURELISE.MAT contient les signaux enregistrés d'un morceau de piano bien connu.

OT SIGNAL
« Analyse du signal de parole »
Travaux Pratiques

1 INTRODUCTION

L'objectif de ce TP est de vous permettre d'appréhender les propriétés du signal de parole tout en découvrant des outils de base de MATLAB pour le traitement du signal comme **Sptool** (cf. annexe).

2 ANALYSE DU SIGNAL DE PAROLE

2.1 Magnétophone Windows

Lancez le magnétophone de Windows NT :

`Démarrer/Programmes/Accessoires/Multimédia/Magnétophone`

Familiarisez vous avec les menus.

☺ 1) Quels sont les différents formats proposés pour sauvegarder le signal ?

☺ 2) Quels sont la fréquence d'échantillonnage maximale proposée et le pas de quantification le plus fin ? Quels sont les formats qui les proposent ?

On utilisera comme format d'enregistrement par défaut le PCM, Mono, 44 kHz, 16 bits. Le signal obtenu dans ce cas sera considéré comme une reproduction parfaite du signal analogique.

☺ 3) Avec ce format, pour 4s de parole, quel seront le nombre d'échantillons et la taille du fichier associé ? Vérifiez votre calcul en vous enregistrant.

2.2 Le son sous MATLAB

Lancez MATLAB. Copiez `tc-nt/partage/TP-Parole` dans un répertoire de travail local.
Placez vous (`cd nom du répertoire`) dans votre répertoire de travail.
Dans la fenêtre de commande de MATLAB, tapez `global sigsample.` (le rôle de cette variable est défini ci-dessous, paragraphe 2.3).
Enregistrez avec le magnétophone une phrase (<5s) et sauvez la en .wav dans votre répertoire de travail.
Dans la fenêtre de commande de MATLAB, tapez :

```
[sig,fe,nb]=worksound('votrephrase.wav') ;
```

☺ 4) Editez `worksound.m`, et étudiez les fonctions MATLAB qui permettent de lire, sauvegarder et émettre un son. On utilisera la fonction `help` pour connaître le rôle et les paramètres de ces fonctions.

Lancez MATLAB. Copiez

`\partagePourLenseignant\Telecoms\TP-Parole\partie1`

dans un répertoire de travail local.
Placez vous (`cd nom du répertoire`) dans votre répertoire de travail.

2.3 Les phonèmes

Nous allons étudier les propriétés des principaux types de phonèmes qui composent le signal de parole.
Dans la fenêtre de commande de MATLAB, lancez `sptool`. Dans la fenêtre `sptool`, ouvrez la session `ParoleSession` qui se trouve dans votre répertoire de travail. Un certain nombre de sons, de filtres et de spectres vous sont donnés.

☺ 5) Observez un des filtres qui vous sont donnés. Quelles sont les propriétés de ce filtre ?

Pour information, les autres filtres ont des paramètres identiques, exceptée la bande passante.
Toute cette partie est faite en utilisant `sptool`, le `signal browser`, et le `spectre viewer`.

2.3.1 Voyelles et sons voisés

☺ 6) Comparez les sons `son_o` et `son_e`.

Pour cela on regardera les signaux temporels et leurs spectres. On repérera le fondamental (pitch), les harmoniques, la position des formants.
Pour les spectres, on utilisera la méthode de `Welch` avec `Nfft=4096`, `Nwind=4096`, `Overlap=2048`. On pensera à utiliser `inherit` pour paramétrer les spectres suivants !

☺ 7) Filtrage des sons `son_o` et `son_e`. Filtrer ces deux sons à respectivement 8, 4, 2, 1, 0.5 kHz. Ecoutez (si vous pouvez :-) et commentez.

☺ 8) Filtrage du son `son_i`. Filtrer ce son à 0.5 kHz. Ecoutez (si vous pouvez :-) et commentez.

2.3.2 Occlusives

☺ 9) Comparez les sons `son_be` et `son_pe`.

☺ 10) Transformez par une opération adéquate le `b` en `p` !

2.3.3 Fricatives

☺ 11) Comparez les sons `son_sse` et `son_zze`. Tronquez les signaux temporels pour récupérer la partie qui correspond aux sons fricatifs. Comparez les spectres des sons fricatifs purs.

2.3.4 Phrase

☺ 12) Visualisez et écoutez le signal `phrase_exemple`. Repérez les principaux phonèmes.

☺ 13) Filtrez cette phrase à respectivement 8, 4, 2, 1, 0.5 kHZ. Ecoutez et commentez. Quels sons disparaissent en premier ? Pourquoi ?

☺ 14) Superposez la phrase initiale avec la phrase filtrée à 1 kHz. Indiquez les phonèmes disparus.

ANNEXE : SPTOOL

1) Synthèse de filtres

- Dans la fenêtre *SPTool,* Cliquez sur 'new design'. Une fenêtre *Filter Designer* s'ouvre.
- Saisissez le type de filtre, puis le gabarit (Fp,Rp,Fs,Rs,Sampling Frequency) et cliquez sur 'apply'.
- Puis cliquez sur 'File / Close'. Votre filtre synthétisé apparaît avec le nom filt1, donnez lui un nom par 'Edit / Name'
- Sauvegardez votre session avec 'File / Save session'. (Faites le de temps en temps)

2) Visualisation des caractéristiques du filtres

- Dans la fenêtre *SPTool*, choisissez un filtre, puis cliquez sur 'View'. Une fenêtre *Filter Viewer* s'ouvre.
- Dans la fenêtre *Filter Viewer*, sélectionnez les réponses que vous voulez observer (zéros/pôles, module, phase, retard de groupe, impulsionnelle, indicielle ...) et les échelles à utiliser. Zoomez, explorez.

3) Visualisation des signaux

- Dans la fenêtre de commande de matlab : Créer un vecteur w de 0 à 4pi sur 1024 points. A partir de là, créer deux vecteurs sin et sincard respectivement égaux à sin(w) et sinc(w).
- Dans la fenêtre *SPTool,* importez ces différents signaux avec 'File / Import'.
- Choisissez en un ou plusieurs, cliquez sur 'View' et explorez la nouvelle fenêtre *Signal Browser*. (Mettez le casque :-)
- Pour faire un 'play' d'un morceau de signal, il faut positionner les 'markers' pour delimiter la partie à jouer.

4) Filtrage

- Dans la fenêtre *SPTool,* sélectionnez un signal et un filtre, puis cliquez sur 'Apply'.
- Avec le *Signal Browser*, observez le signal avant et après filtrage.

5) Analyse spectrale d'un signal

- Dans la fenêtre *SPTool,* sélectionnez un signal, puis dans la colonne Spectra, cliquez sur 'Create'.
- Dans la fenêtre *Spectrum Viewer* qui vient d'apparaître choississez une méthode d'analyse spectrale, puis 'Apply'. Cela vous donne le spectre du signal.

6) Export

Vous pouvez renvoyer sous la fenêtre de commande de matlab ce que vous avez créé sous la fenêtre SPTools avec 'File / Export'. Cette commande crée dans la fenêtre de commande une variable du nom de l'objet exporté.
Cette variable est une structure. Tapez son nom suivi de 'enter', vous verrez la liste des champs de la structure apparaître.

Par exemple pour un filtre appelé B. En tapant B.tf.num, vous accédez au numérateur de la fonction de transfert en Z, B.tf.den et c'est le dénominateur ...

OT SIGNAL
« Introduction à SIMULINK »
Travaux Pratiques

1. INTRODUCTION

L'objectif de ce TP est de découvrir et d'utiliser un outil professionnel de simulation de systèmes numériques intégré à l'environnement Matlab : SIMULINK.

2. SIMULINK : Principes de base

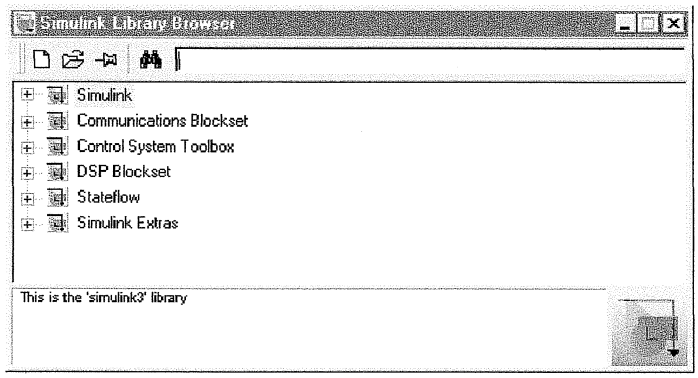
SIMULINK est un programme additionnel à MATLAB pour la simulation de systèmes dynamiques. Il permet, par exemple, de simuler complètement le fonctionnement interne d'un modem et d'étudier le comportement de chacun de ses blocs.
Le contrôle et la modélisation des systèmes deviennent plus aisés; les fonctions de transfert sont écrites sous forme de blocs et les liaisons sont réalisées par des arcs orientés. Les différents types de signaux peuvent être générés et visualisés à l'aide d'instruments virtuels.
Un modèle construit à l'aide d'un assemblage de blocs élémentaires peut être encapsulé. Il peut alors enrichir la bibliothèque disponible sous SIMULINK. Le système modélisé sous SIMULINK peut recevoir des données de l'espace de travail de MATLAB ou y envoyer des données de sortie. L'échange de données entre SIMULINK et l'espace de travail de MATLAB peut se faire par des variables communes ou par des fichiers *.mat*.

2.1 Les librairies SIMULINK

SIMULINK peut être invoqué sous MATLAB par :

```
>> simulink
```

On aboutit à la fenêtre 'browser' suivante qui permet d'accéder au différentes librairies simulink installées.



On peut également à partir de cette fenêtre créer un nouveau schéma simulink ou en ouvrir un existant.
Un double click sur simulink fait apparaître les librairies suivantes :

- `sources` : générateurs de signaux (carré, sinus, ...)

- `sinks` : contient les blocs permettant de diriger des valeurs vers un oscilloscope, un fichier, ou l'espace de travail.
- `discrete` : modèles discrets
- `continuous` : modèles analogiques linéaires
- `non-linear` : non linéarités (seuils, relais, ...)
- ...

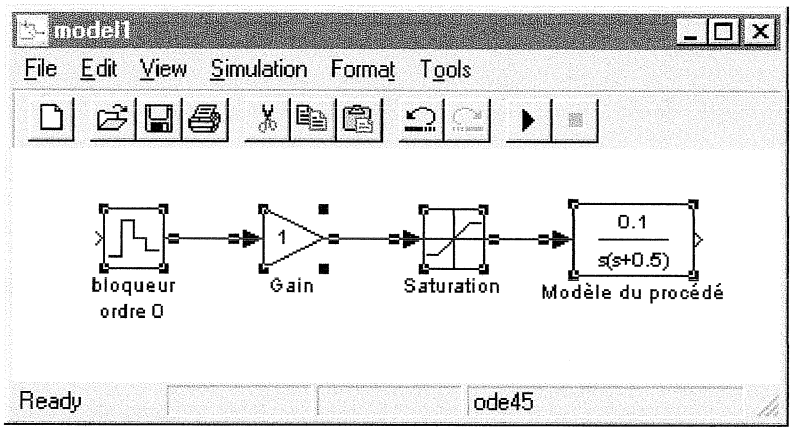
En sélectionnant par un double clic, le bloc `sources`, on ouvre la librairie des sources. Cette fenêtre contient des blocs qui vont permettre de simuler différents types de sources.

On peut ouvrir un nouveau schéma *simulink*, et on aboutit à une deuxième fenêtre vide nommée '*untitled*' qui est la fenêtre dans laquelle sera conçu et visualisé le système à étudier.
On peut cliquer sur le générateur de sinusoïde et en maintenant appuyé aller le poser sur la fenêtre '*untitled*'. Là, un double clic sur le bloc posé permet d'accéder à la configuration du générateur de sinusoïdes.

2.2 Encapsulation (Masquage)

L'encapsulation ou masquage consiste à transformer un ensemble de blocs en un seul. Ce dernier aura ensuite les mêmes propriétés que les blocs élémentaires des librairies SIMULINK.

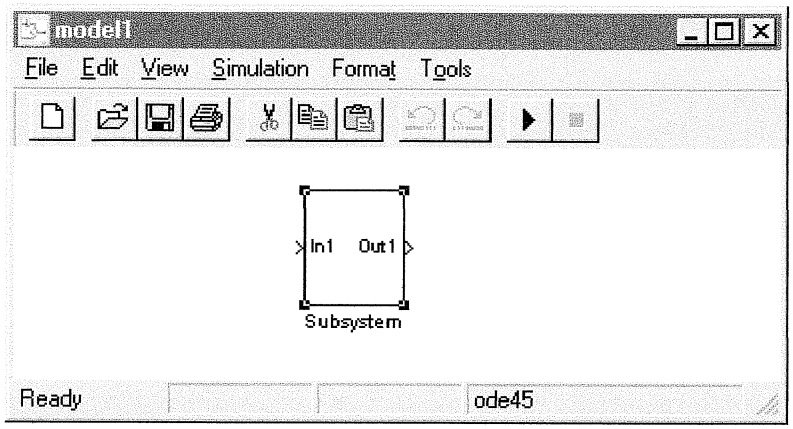
Exemple :
Considérons par exemple, un modèle de processus analogique précédé d'un bloqueur d'ordre zéro. Le procédé est attaqué par un amplificateur de puissance de gain K dont la tension de sortie est limitée entre les valeurs -V et +V. Dans une nouvelle fenêtre, on assemble les différents blocs nécessaires que l'on extrait des librairies SIMULINK.



Pour réaliser ce système, il va falloir incorporer dans la fenêtre *'untitled'* quatre blocs, les configurer et enfin les relier entre eux.

- discrete / zero-holder hold
- linear / gain
- non-linear / saturation
- linear : zero-pole

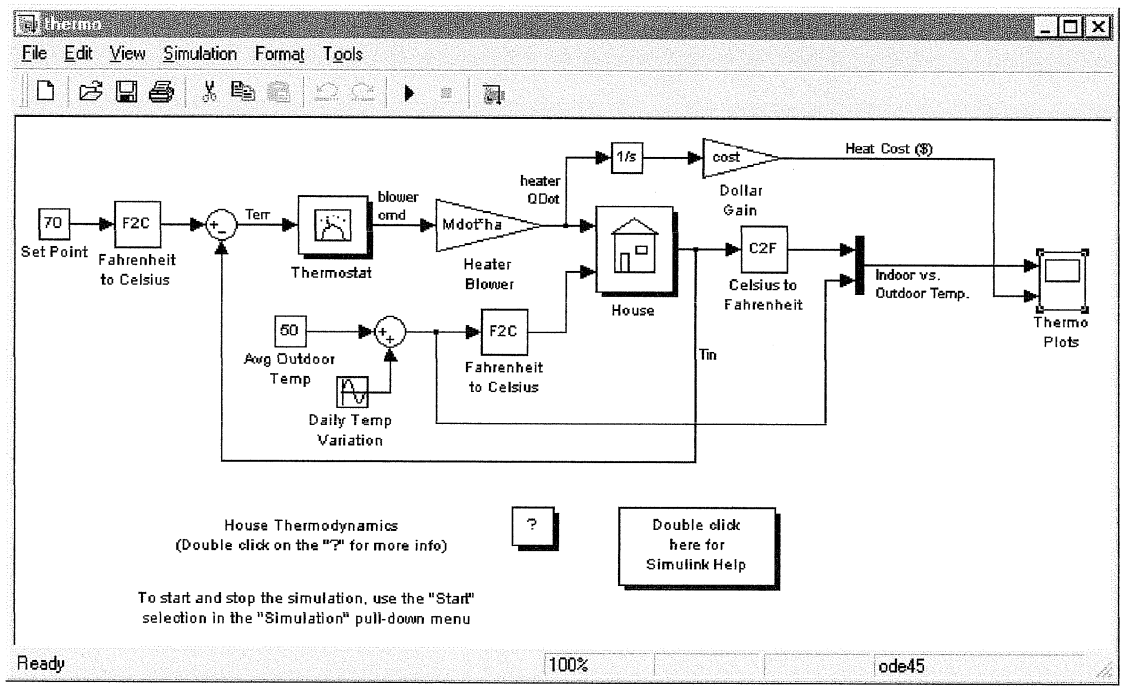
Une fois ces différents blocs reliés entre eux, on sélectionne l'ensemble à la souris et on les groupe à l'aide de la commande Edit/CreateSubsystem; On obtient alors le bloc suivant :



Un double clic sur le bloc que l'on vient de créer permet de retrouver les icônes assemblés auxquels SIMULINK a automatiquement ajouté un bloc d'entrée *In1* et un bloc de sortie *Out1*. En double cliquant sur l'un d'entre eux, on a accès aux paramètres du sous-système.

2.3 Une petite démonstration

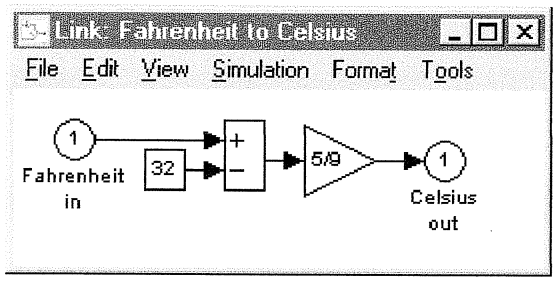
Dans la fenêtre MATLAB, tapez *thermo*. Cette commande lance SIMULINK et fait apparaître la fenêtre suivante qui contient un modèle de la thermodynamique d'une maison.



Un double click sur le bloc *thermo plot* fait apparaître une fenêtre de visualisation. Lancer la simulation par *Simulation/Start* ou en cliquant sur le bouton ▶. Pendant l'exécution des signaux apparaissent dans la fenêtre 'thermo plots'.

Ce modèle inclue plusieurs sous-systèmes afin d'en faciliter la lisibilité : les deux convertisseurs de température (F2C et C2F), le thermostat, la maison. Certains sont

masqués, pour en voir le contenu, il faut sélectionner le sous-système, appeler un menu avec le bouton droit de la souris et choisir *Look under mask.*
Les sous-systèmes comme le thermostat, s'ouvre par un double clic, comme le thermostat dont le détail est donné ci-dessous.



2.3.1 Quelques petites manipulations

• Nous allons augmenter la durée de la simulation pour pouvoir réaliser des opérations en cours de simulation.
Appelez le menu *Simulation/Parameters* et modifiez le *stop-time*. Il est donné en secondes et initialement mis à 2 jours. Passez le à 40 jours.

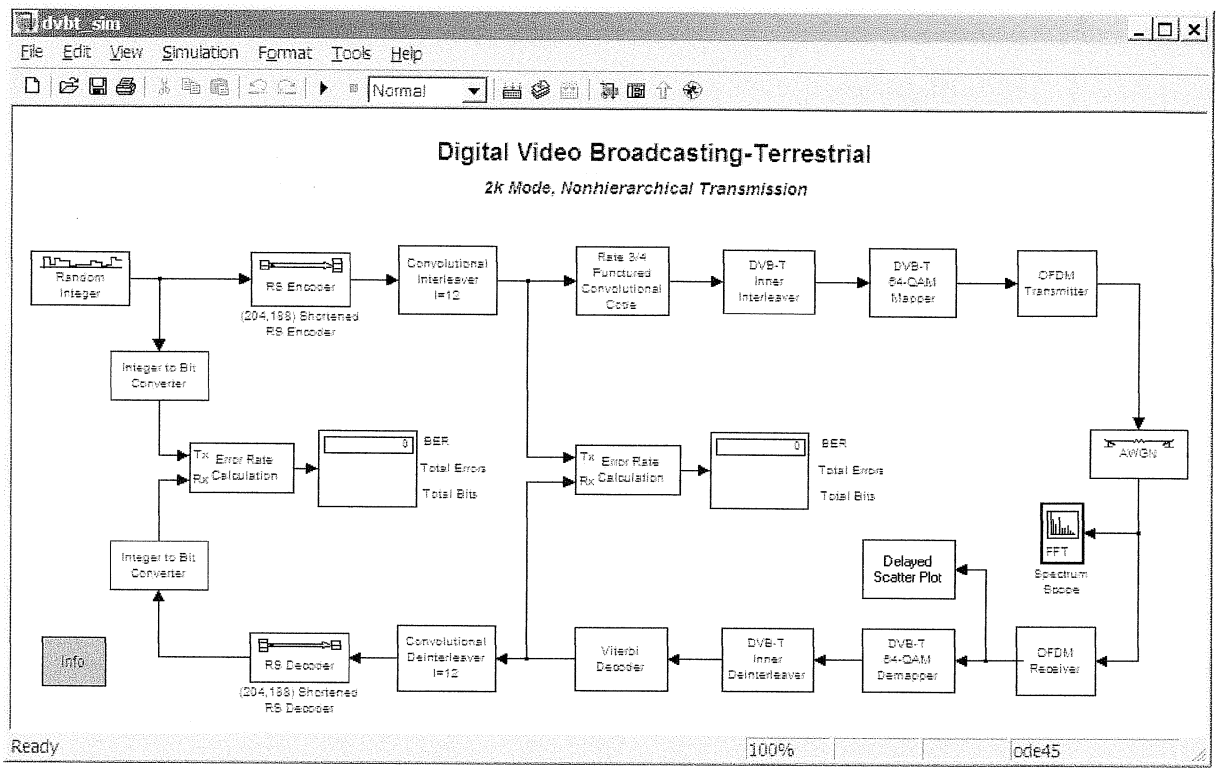
• Nous allons ajouter un oscilloscope 'flottant' qui va nous permettre d'observer n'importe lequel des liens (1 lien = 1 signal) du modèle pendant la simulation.
Faites glissez le bloc *Scope* de la fenêtre de la librairie *Simulink/Sinks* sur la fenêtre du modèle *thermo*.
Dans la fenêtre *thermo*, double cliquez sur le bloc *Scope*. Une fenêtre s'ouvre. Dans cette fenêtre cliquez sur l'icône *Properties* et cochez *Floating scope*.
Lancez la simulation. Sur le modèle sélectionnez un lien ; dans la fenêtre de l'oscillo flottant cliquez sur l'icône en forme de jumelles et observez. Vous pouvez répétez cette opération sur un autre lien sans arrêter la simulation.

• Nous allons modifier des paramètres du modèle au cours de la simulation.
Lancez la simulation et double cliquez sur la sinusoïde qui simule les variations de température et modifiez l'amplitude. Observez ce qui se passe sur les oscilloscopes. Puis juste après, modifiez en double cliquant sur 50 la valeur moyenne de la température extérieure et observez !

2.4 Une grosse démonstration

Dans la fenêtre MATLAB, tapez *dvbt_sim*. Cette commande lance SIMULINK et fait apparaître la fenêtre suivante qui contient un modèle de transmission en télévision numérique terrestre.

Analysez le schéma et repérez la partie émission, réception, le canal et les éléments qui vont être visualisés.
Lancez la simulation, et observez.



Ces premières manipulations illustrent la puissance d'un tel outil. Sans compter l'aspect interface qui se découvre naturellement puisque par quelques clics intuitifs vous pouvez changer le nom des boites, coloriez les éléments du modèle ...

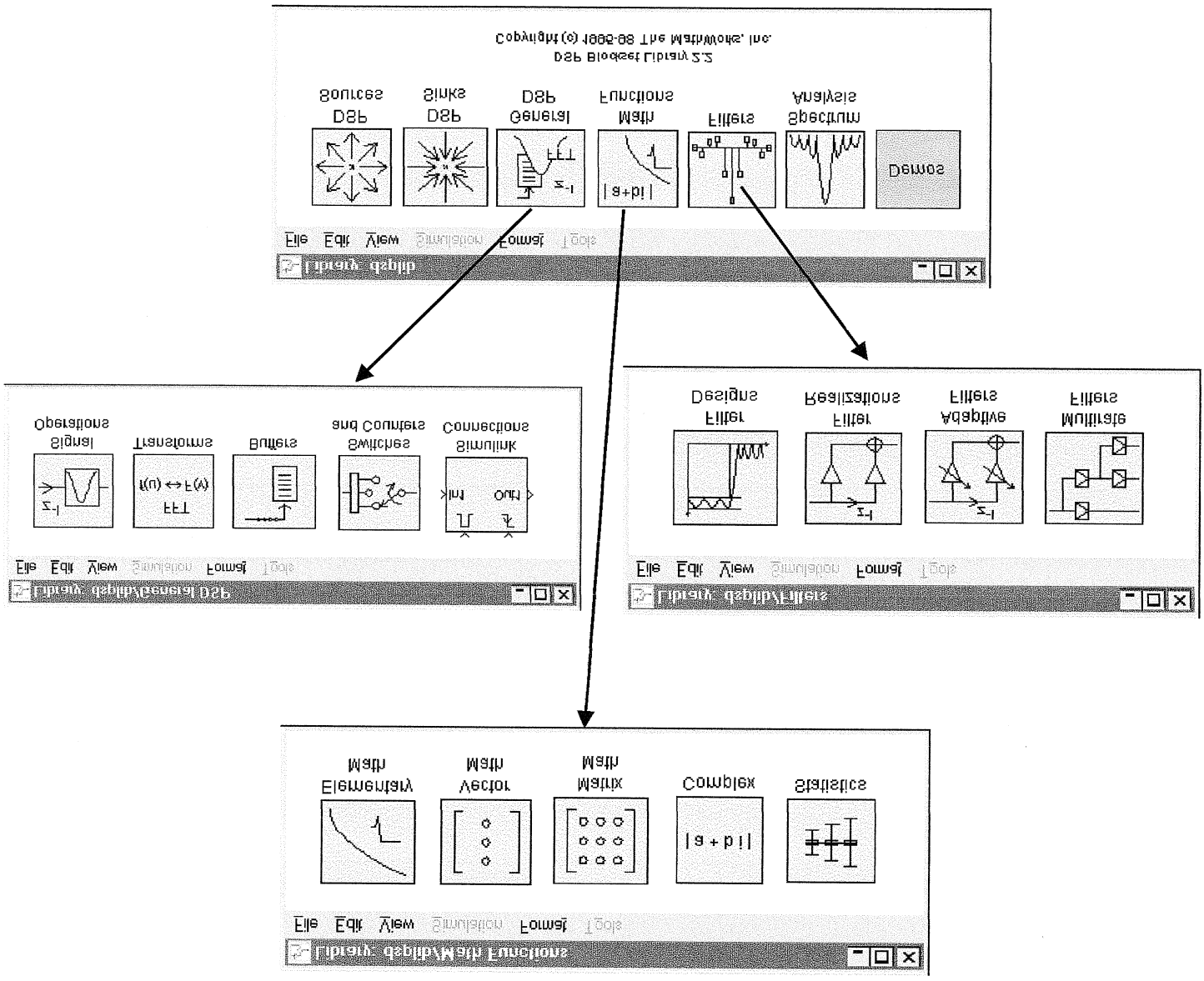
3. Le DSP Blockset

Comme MATLAB possède des Toolbox (Signal, Image, Communication, ...), SIMULINK possède des Blockset, c'est-à-dire des librairies de blocs prédéfinis et prêts à l'emploi. Nous nous intéresserons ici au DSP (Digital Signal Processing) Blockset qui regroupe des blocs équivalents aux fonctions du Toolbox Signal Processing.

3.1 Vue d'ensemble

Pour accéder à cette librairie, cliquez sur le bloc *Blocksets & Toolboxes* dans la fenêtre *Library:Simulink*, puis sur le bloc *DSP Blockset* dans la fenêtre qui apparaît.
La fenêtre *Library:dsplib* apparaît. Elle permet d'accéder aux blocs du DSP Blockset. Cette librairie est divisée en 6 sous-ensembles comme le montre le schéma suivant.

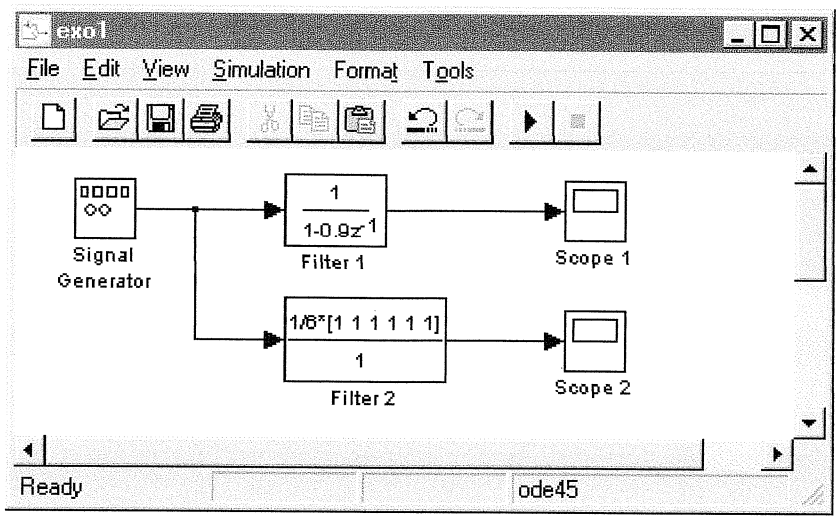
Pour accéder au contenu de chaque ensemble, il suffit de cliquer dessus. Les sous-ensembles *General DSP*, *Math Functions* et *Filters* sont eux même divisés en sous-ensembles. On pourra étudier le contenu de *General DSP*.



3.2 Construisez votre premier modèle

A partir de la fenêtre browser, ouvrez un nouveau modèle. Une Fenêtre '*untitled*' apparaît. C'est là que vous allez concevoir votre modèle et le simuler. Pensez à sauver régulièrement votre modèle par *File/Save*.

Vous allez concevoir le modèle suivant :



Pour cela, vous allez devoir aller chercher 3 blocs :

- *Simulink/Sources/Signal Generator*
- *Simulink/Sinks/Scope*
- *Simulink/Discrete/Discrete Filter*

Vous allez dupliquer les blocs *Filter* et *Scope* par un ^C ^V.
Vous allez les relier entre eux. Pour créer un lien, on utilise le bouton gauche de la souris. Pour effectuer un branchement sur un lien existant, il faut sélectionner le lien et utiliser le bouton droit de la souris.
Vous allez ensuite paramétrer les différents blocs en double cliquant dessus :

- On choisira un signal sinusoïdal d'amplitude 1 et de fréquence 2,5 Hz
- On définira les deux filtres en choisissant pour *Sample Time* : 0.01.
- En cliquant sur les scopes, on fait apparaître les fenêtres de visualisation. On explorera les rôles des différents boutons.

On va ensuite paramétrer la simulation par *Simulation/Parameters* en choisissant un *Stop Time* de 4s.

Vous pouvez lancer la simulation par *Simulation/Start* ou en cliquant sur le bouton ▶|.

OT Bases du signal

Séance machine n°3

Résumé
On propose l'analyse en temps et en fréquence d'un signal composé de 2 impulsions gaussiennes de fréquence centrale et de bande passante différentes. Après avoir mesurer les caractéristiques en temps et en fréquence des signaux, la notion de filtrage sera introduite dans le but d'identifier les fréquences appartenant à chacune des impulsions gaussiennes.

Canevas de la séance

1 Analyse d'impulsions gaussiennes en temps et en fréquence

- Générer une impulsion gaussienne d'équation $x(n) = Ae^{-\pi B^2(n-n_0)^2}$, $n \in [0, N-1]$, et calculer son spectre $X(f) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi fn}$, $f \in [0,1[$, avec

  A l'amplitude, 1/B la demie largeur de l'impulsion à mi-hauteur, n_0 le centre de l'impulsion, N le nombre de points

  Choisir les paramètres suivants puis les modifier pour visualiser les effets sur le signal et son spectre.
  A1 = 0.5; B1 = 0.05; n1 = 40;

- Générer une impulsion gaussienne modulée par une porteuse sinusoïdale. L'équation est :
  $x(n) = Ae^{-\pi B^2(n-n_0)^2} \cos(2\pi f_0(n-n_0))$, $n \in [0, N-1]$, et calculer son spectre
  $X(f) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi fn}$, $f \in [0,1[$, avec f_0 la fréquence de la porteuse.
  Fixer la fréquence porteuse à $f_1 = 0.1$ puis modifier cette valeur pour visualiser les effets sur le signal et son spectre.

- Générer une somme de 2 impulsions gaussiennes modulées $x'(n)$ perturbées par un bruit blanc gaussien additif $b(n)$.
  $x'(n) = x_1(n) + x_2(n)$
  Les valeurs des paramètres du signal $x_1(n)$ sont celles du paragraphe précédent. Les valeurs pour $x_2(n)$ sont : A2 = 1; B2 = 0.1; n2 = 80; f2 = 0.2
  Le signal $x(n)$ est :
  $x(n) = x'(n) + \sigma\, b(n)$ avec $\sigma^2 = \frac{10^{-RSB/10}}{N} \sum_{n=0}^{N-1} x'^2(n)$
  Observer le signal $x(n)$ et son spectre pour différentes valeurs du paramètre RSB (Rapport Signal à Bruit) exprimé en décibel. On pourra faire varier ce paramètre entre -10 dB et +40 dB par exemple.

- Calculer le centroïde du spectre du signal $x(n)$ en estimant la fréquence moyenne à partir de la densité spectrale d'énergie du signal $P_{xx}(f) = |X(f)|^2$.

$$\bar{f} = \frac{\int_{-\infty}^{+\infty} f P_{xx}(f) df}{\int_{-\infty}^{+\infty} P_{xx}(f) df}$$

Pour obtenir la fréquence centrale de chaque impulsion gaussienne, il est nécessaire de séparer leur contenu spectral. Cette séparation s'obtient par filtrage.

2 Filtrage des impulsions gaussiennes

Toutes les étapes décrites ci-dessous seront à réaliser deux fois. Une première fois pour isoler l'impulsion basse fréquence, puis une seconde fois, pour isoler l'impulsion haute fréquence.

- Conception d'un filtre
  On propose d'utiliser la méthode de la fenêtre pour concevoir le filtre. Cette méthode est mise en œuvre dans la fonction *fir1* de Matlab. On choisira un nombre de coefficients de $N_c = 40$ et une fréquence de coupure $f_c = 0.15$ de façon à isoler l'impulsion basse fréquence.
  Observer la réponse impulsionnelle du filtre et son spectre.

- Application du filtre
  Le signal filtré $y(n)$ s'obtient en calculant l'opération de convolution entre le signal non filtré $x(n)$ et la réponse impulsionnelle du filtre $h(n)$. Ce calcul est mis en œuvre dans la fonction Matlab *filter* et s'exprime par l'équation : $y(n) = \sum_{i=0}^{N_c-1} h(i)x(n-i)$
  Observer le signal filtré et son spectre.

- Calcul du centroïde du spectre du signal $y(n)$

ANALYSE DE FOURIER A COURT TERME

Didier VRAY- CREATIS - INSA - Lyon

Le but du travail est d'analyser le signal contenu dans le fichier **tone1.mat.** L'objectif est de décoder ce signal qui est constitué d'une succession de sinusoïdes de fréquence constante sur une courte durée. Le signal inconnu résulte de l'encodage d'une séquence de digits représentant des fréquences différentes suivant la correspondance suivante :

Digit	0	1	2	3	4	5	6	7	8	9
F(Hz)	250	750	1250	1750	2250	2750	3250	3750	4250	4750

Le signal est échantillonné à la fréquence f = 10kHz.

1 - Lire le fichier avec la commande **Load** de Matlab.
Représenter le signal. Graduer correctement l'axe temporel.
Quel est le nombre d'échantillons du signal? Quelle est sa durée? Repérer les différentes composantes temporelles du signal. Evaluer visuellement la durée de ces composantes.

2 - Calculer la Transformée de Fourier du signal. Choisir le nombre de points de la transformée de Fourier. Représenter le spectre fréquentiel correspondant aux fréquences positives. Graduer correctement l'axe fréquentiel. Analyser le contenu fréquentiel. Combien de composantes spectrales peut-on mettre en évidence. A quelle fréquence correspond chacune d'entre elles.

3 - Calculer séparément la Transformée de Fourier de plusieurs tranches du signal. Montrer que chacune de ces tranches n'a pas le même contenu fréquentiel.

4 - Il s'agit maintenant de décoder la suite de digits codés en fréquence en prenant successivement des morceaux du signal. Proposer un programme qui permet le décodage de la séquence en calculant la Transformée de Fourier de chaque tranche et en détectant le pic de fréquence. Pour cela, le signal sera découpé "manuellement", les numéros des échantillons de début et de fin de chaque tranche seront fournis au programme suite à l'analyse temporelle précédente.

5 - Proposer une analyse temps-fréquence par le spectrogramme.

L'expression du spectrogramme est : $S_x(t,f;h)=\left|\int_{-\infty}^{+\infty} x(\tau)h^*(\tau-t)e^{-j2\pi f\tau}d\tau\right|^2$

x est le signal. h est la fenêtre d'analyse de durée limitée. Le résultat est présenté sous la forme d'une image avec l'axe temporel sur l'horizontal, l'axe fréquentiel sur la verticale. Chaque pixel est codé en couleur en fonction de l'amplitude de $S_x(t,f)$.
On pourra utiliser la fonction Specgram de Matlab qui calcule et affiche le spectrogramme du signal x : specgram(x,Nfft,Fe,FenetreAnalyse, Ndecal)

Nfft est le nombre de points de la Transformée de Fourier
Fe est la fréquence d'échantillonnage
FenetreAnalyse est la fenêtre d'analyse de longueur N points
Ndecal est le nombre de points de décalage entre les fenêtres d'analyse

Retrouver le code de la séquence à partir de l'image temps-fréquence. Justifier le choix des paramètres.

6 - D'autres signaux fournis peuvent être analysés avec le spectrogramme :
Le fichier VOWELS.MAT contient les sons des voyelles AEIOU parlées successivement (en anglais).
B3PULSES.MAT contient des impulsions gaussiennes. Il s'agit de les localiser en temps et en fréquence.
Le fichier FURELISE.MAT contient les signaux enregistrés d'un morceau de piano bien connu.

OT SIGNAL
« Analyse du signal de parole »
Travaux Pratiques

1 INTRODUCTION

L'objectif de ce TP est de vous permettre d'appréhender les propriétés du signal de parole tout en découvrant des outils de base de MATLAB pour le traitement du signal comme **Sptool** (cf. annexe).

2 ANALYSE DU SIGNAL DE PAROLE

2.1 Magnétophone Windows

Lancez le magnétophone de Windows NT :

```
Démarrer/Programmes/Accessoires/Multimédia/Magnétophone
```

Familiarisez vous avec les menus.

☺ 1) Quels sont les différents formats proposés pour sauvegarder le signal ?

☺ 2) Quels sont la fréquence d'échantillonnage maximale proposée et le pas de quantification le plus fin ? Quels sont les formats qui les proposent ?

On utilisera comme format d'enregistrement par défaut le PCM, Mono, 44 kHz, 16 bits. Le signal obtenu dans ce cas sera considéré comme une reproduction parfaite du signal analogique.

☺ 3) Avec ce format, pour 4s de parole, quel seront le nombre d'échantillons et la taille du fichier associé ? Vérifiez votre calcul en vous enregistrant.

2.2 Le son sous MATLAB

Lancez MATLAB. Copiez `tc-nt/partage/TP-Parole` dans un répertoire de travail local.
Placez vous (`cd nom du répertoire`) dans votre répertoire de travail.
Dans la fenêtre de commande de MATLAB, tapez `global sigsample.` (le rôle de cette variable est défini ci-dessous, paragraphe 2.3).
Enregistrez avec le magnétophone une phrase (<5s) et sauvez la en .wav dans votre répertoire de travail.
Dans la fenêtre de commande de MATLAB, tapez :

```
[sig,fe,nb]=worksound('votrephrase.wav') ;
```

☺ 4) Editez `worksound.m`, et étudiez les fonctions MATLAB qui permettent de lire, sauvegarder et émettre un son. On utilisera la fonction `help` pour connaître le rôle et les paramètres de ces fonctions.

Lancez MATLAB. Copiez

```
\partagePourLenseignant\Telecoms\TP-Parole\partie1
```

dans un répertoire de travail local.
Placez vous (`cd nom du répertoire`) dans votre répertoire de travail.

2.3 Les phonèmes

Nous allons étudier les propriétés des principaux types de phonèmes qui composent le signal de parole.
Dans la fenêtre de commande de MATLAB, lancez `sptool`. Dans la fenêtre `sptool`, ouvrez la session `ParoleSession` qui se trouve dans votre répertoire de travail. Un certain nombre de sons, de filtres et de spectres vous sont donnés.

☺ 5) Observez un des filtres qui vous sont donnés. Quelles sont les propriétés de ce filtre ?

Pour information, les autres filtres ont des paramètres identiques, exceptée la bande passante.
Toute cette partie est faite en utilisant `sptool`, le `signal browser`, et le `spectre viewer`.

2.3.1 Voyelles et sons voisés

☺ 6) Comparez les sons `son_o` et `son_e`.

Pour cela on regardera les signaux temporels et leurs spectres. On repérera le fondamental (pitch), les harmoniques, la position des formants.
Pour les spectres, on utilisera la méthode de `Welch` avec `Nfft=4096`, `Nwind=4096`, `Overlap=2048`. On pensera à utiliser `inherit` pour paramétrer les spectres suivants !

☺ 7) Filtrage des sons `son_o` et `son_e`. Filtrer ces deux sons à respectivement 8, 4, 2, 1, 0.5 kHz. Ecoutez (si vous pouvez :-) et commentez.

☺ 8) Filtrage du son `son_i`. Filtrer ce son à 0.5 kHz. Ecoutez (si vous pouvez :-) et commentez.

2.3.2 Occlusives

☺ 9) Comparez les sons `son_be` et `son_pe`.

☺ 10) Transformez par une opération adéquate le b en p !

2.3.3 Fricatives

☺ 11) Comparez les sons `son_sse` et `son_zze`. Tronquez les signaux temporels pour récupérer la partie qui correspond aux sons fricatifs. Comparez les spectres des sons fricatifs purs.

2.3.4 Phrase

☺ 12) Visualisez et écoutez le signal `phrase_exemple`. Repérez les principaux phonèmes.

☺ 13) Filtrez cette phrase à respectivement 8, 4, 2, 1, 0.5 kHZ. Ecoutez et commentez. Quels sons disparaissent en premier ? Pourquoi ?

☺ 14) Superposez la phrase initiale avec la phrase filtrée à 1 kHz. Indiquez les phonèmes disparus.

ANNEXE : SPTOOL

1) Synthèse de filtres

- Dans la fenêtre *SPTool,* Cliquez sur 'new design'. Une fenêtre *Filter Designer* s'ouvre.
- Saisissez le type de filtre, puis le gabarit (Fp,Rp,Fs,Rs,Sampling Frequency) et cliquez sur 'apply'.
- Puis cliquez sur 'File / Close'. Votre filtre synthétisé apparaît avec le nom filt1, donnez lui un nom par 'Edit / Name'
- Sauvegardez votre session avec 'File / Save session'. (Faites le de temps en temps)

2) Visualisation des caractéristiques du filtres

- Dans la fenêtre *SPTool*, choisissez un filtre, puis cliquez sur 'View'. Une fenêtre *Filter Viewer* s'ouvre.
- Dans la fenêtre *Filter Viewer*, sélectionnez les réponses que vous voulez observer (zéros/pôles, module, phase, retard de groupe, impulsionnelle, indicielle ...) et les échelles à utiliser. Zoomez, explorez.

3) Visualisation des signaux

- Dans la fenêtre de commande de matlab : Créer un vecteur w de 0 à 4pi sur 1024 points. A partir de là, créer deux vecteurs sin et sincard respectivement égaux à sin(w) et sinc(w).
- Dans la fenêtre *SPTool,* importez ces différents signaux avec 'File / Import'.
- Choisissez en un ou plusieurs, cliquez sur 'View' et explorez la nouvelle fenêtre *Signal Browser*. (Mettez le casque :-)
- Pour faire un ‘play’ d’un morceau de signal, il faut positionner les ‘markers’ pour delimiter la partie à jouer.

4) Filtrage

- Dans la fenêtre *SPTool,* sélectionnez un signal et un filtre, puis cliquez sur 'Apply'.
- Avec le *Signal Browser***,** observez le signal avant et après filtrage.

5) Analyse spectrale d'un signal

- Dans la fenêtre *SPTool,* sélectionnez un signal, puis dans la colonne Spectra, cliquez sur 'Create'.
- Dans la fenêtre *Spectrum Viewer* qui vient d'apparaître choisissez une méthode d'analyse spectrale, puis 'Apply'. Cela vous donne le spectre du signal.

6) Export

Vous pouvez renvoyer sous la fenêtre de commande de matlab ce que vous avez créé sous la fenêtre SPTools avec 'File / Export'. Cette commande crée dans la fenêtre de commande une variable du nom de l'objet exporté.
Cette variable est une structure. Tapez son nom suivi de 'enter', vous verrez la liste des champs de la structure apparaître.

Par exemple pour un filtre appelé B. En tapant B.tf.num, vous accédez au numérateur de la fonction de transfert en Z, B.tf.den et c'est le dénominateur ...

OT SIGNAL
« Introduction à SIMULINK »
Travaux Pratiques

1. INTRODUCTION

L'objectif de ce TP est de découvrir et d'utiliser un outil professionnel de simulation de systèmes numériques intégré à l'environnement Matlab : SIMULINK.

2. SIMULINK : Principes de base

SIMULINK est un programme additionnel à MATLAB pour la simulation de systèmes dynamiques. Il permet, par exemple, de simuler complètement le fonctionnement interne d'un modem et d'étudier le comportement de chacun de ses blocs.

Le contrôle et la modélisation des systèmes deviennent plus aisés; les fonctions de transfert sont écrites sous forme de blocs et les liaisons sont réalisées par des arcs orientés. Les différents types de signaux peuvent être générés et visualisés à l'aide d'instruments virtuels.

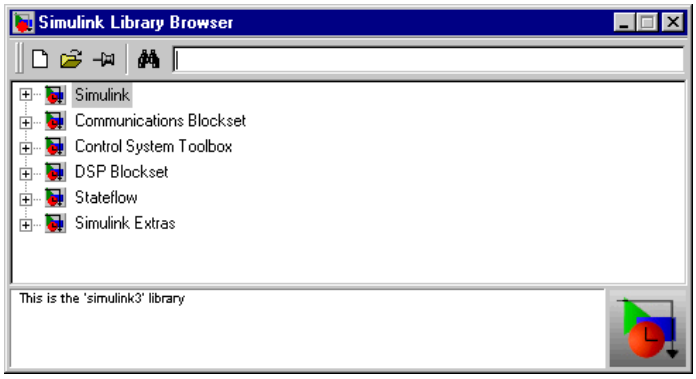
Un modèle construit à l'aide d'un assemblage de blocs élémentaires peut être encapsulé. Il peut alors enrichir la bibliothèque disponible sous SIMULINK. Le système modélisé sous SIMULINK peut recevoir des données de l'espace de travail de MATLAB ou y envoyer des données de sortie. L'échange de données entre SIMULINK et l'espace de travail de MATLAB peut se faire par des variables communes ou par des fichiers *.mat*.

2.1 Les librairies SIMULINK

SIMULINK peut être invoqué sous MATLAB par :

```
>> simulink
```

On aboutit à la fenêtre 'browser' suivante qui permet d'accéder au différentes librairies simulink installées.



On peut également à partir de cette fenêtre créer un nouveau schéma simulink ou en ouvrir un existant.

Un double click sur simulink fait apparaître les librairies suivantes :

- `sources` : générateurs de signaux (carré, sinus, ...)

- `sinks` : contient les blocs permettant de diriger des valeurs vers un oscilloscope, un fichier, ou l'espace de travail.
- `discrete` : modèles discrets
- `continuous` : modèles analogiques linéaires
- `non-linear` : non linéarités (seuils, relais, ...)
- ...

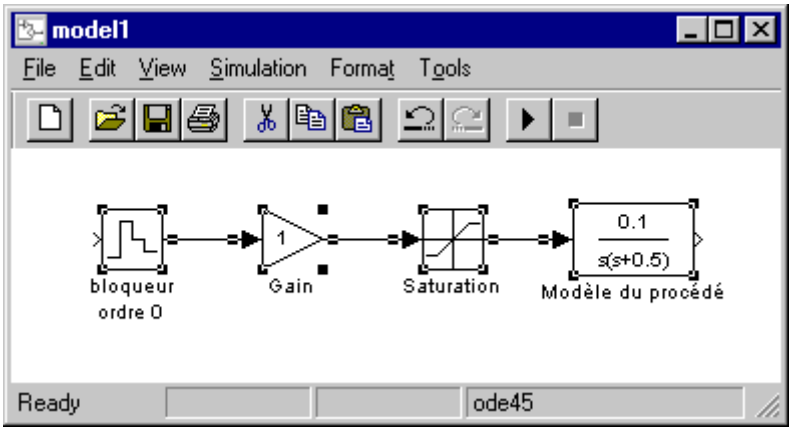
En sélectionnant par un double clic, le bloc `sources`, on ouvre la librairie des sources. Cette fenêtre contient des blocs qui vont permettre de simuler différents types de sources.

On peut ouvrir un nouveau schéma *simulink*, et on aboutit à une deuxième fenêtre vide nommée '*untitled*' qui est la fenêtre dans laquelle sera conçu et visualisé le système à étudier.
On peut cliquer sur le générateur de sinusoïde et en maintenant appuyé aller le poser sur la fenêtre '*untitled*'. Là, un double clic sur le bloc posé permet d'accéder à la configuration du générateur de sinusoïdes.

2.2 Encapsulation (Masquage)

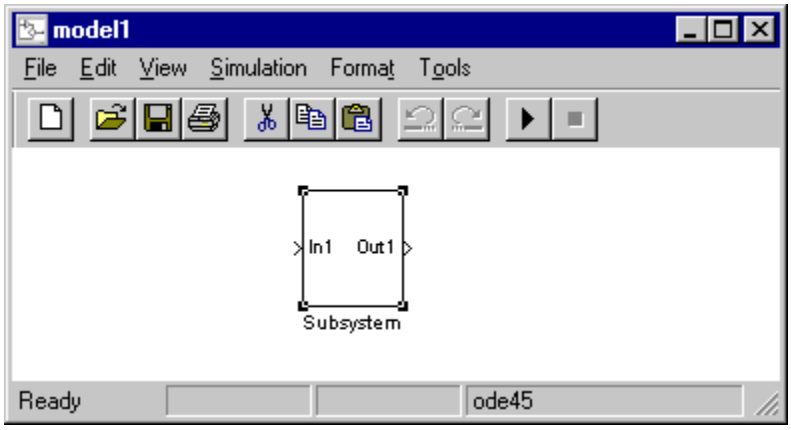
L'encapsulation ou masquage consiste à transformer un ensemble de blocs en un seul. Ce dernier aura ensuite les mêmes propriétés que les blocs élémentaires des librairies SIMULINK.

Exemple :
Considérons par exemple, un modèle de processus analogique précédé d'un bloqueur d'ordre zéro. Le procédé est attaqué par un amplificateur de puissance de gain K dont la tension de sortie est limitée entre les valeurs -V et +V. Dans une nouvelle fenêtre, on assemble les différents blocs nécessaires que l'on extrait des librairies SIMULINK.



Pour réaliser ce système, il va falloir incorporer dans la fenêtre *'untitled'* quatre blocs, les configurer et enfin les relier entre eux.
- discrete / zero-holder hold
- linear / gain
- non-linear / saturation
- linear : zero-pole

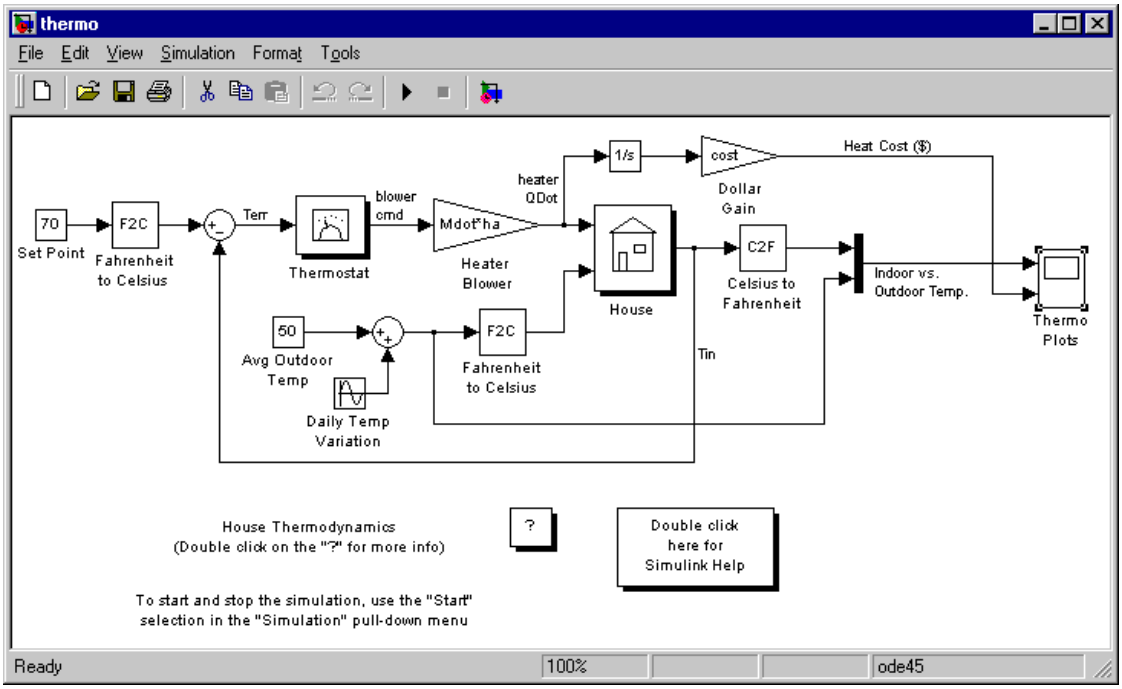
Une fois ces différents blocs reliés entre eux, on sélectionne l'ensemble à la souris et on les groupe à l'aide de la commande Edit/CreateSubsystem; On obtient alors le bloc suivant :



Un double clic sur le bloc que l'on vient de créer permet de retrouver les icônes assemblés auxquels SIMULINK a automatiquement ajouté un bloc d'entrée *In1* et un bloc de sortie *Out1*. En double cliquant sur l'un d'entre eux, on a accès aux paramètres du sous-système.

2.3 Une petite démonstration

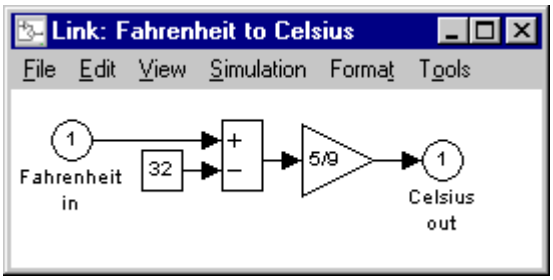
Dans la fenêtre MATLAB, tapez *thermo*. Cette commande lance SIMULINK et fait apparaître la fenêtre suivante qui contient un modèle de la thermodynamique d'une maison.



Un double click sur le bloc *thermo plot* fait apparaître une fenêtre de visualisation. Lancer la simulation par *Simulation/Start* ou en cliquant sur le bouton  . Pendant l'exécution des signaux apparaissent dans la fenêtre 'thermo plots'.

Ce modèle inclue plusieurs sous-systèmes afin d'en faciliter la lisibilité : les deux convertisseurs de température (F2C et C2F), le thermostat, la maison. Certains sont

masqués, pour en voir le contenu, il faut sélectionner le sous-système, appeler un menu avec le bouton droit de la souris et choisir *Look under mask.*
Les sous-systèmes comme le thermostat, s'ouvre par un double clic, comme le thermostat dont le détail est donné ci-dessous.



2.3.1 Quelques petites manipulations

• Nous allons augmenter la durée de la simulation pour pouvoir réaliser des opérations en cours de simulation.
Appelez le menu *Simulation/Parameters* et modifiez le *stop-time*. Il est donné en secondes et initialement mis à 2 jours. Passez le à 40 jours.

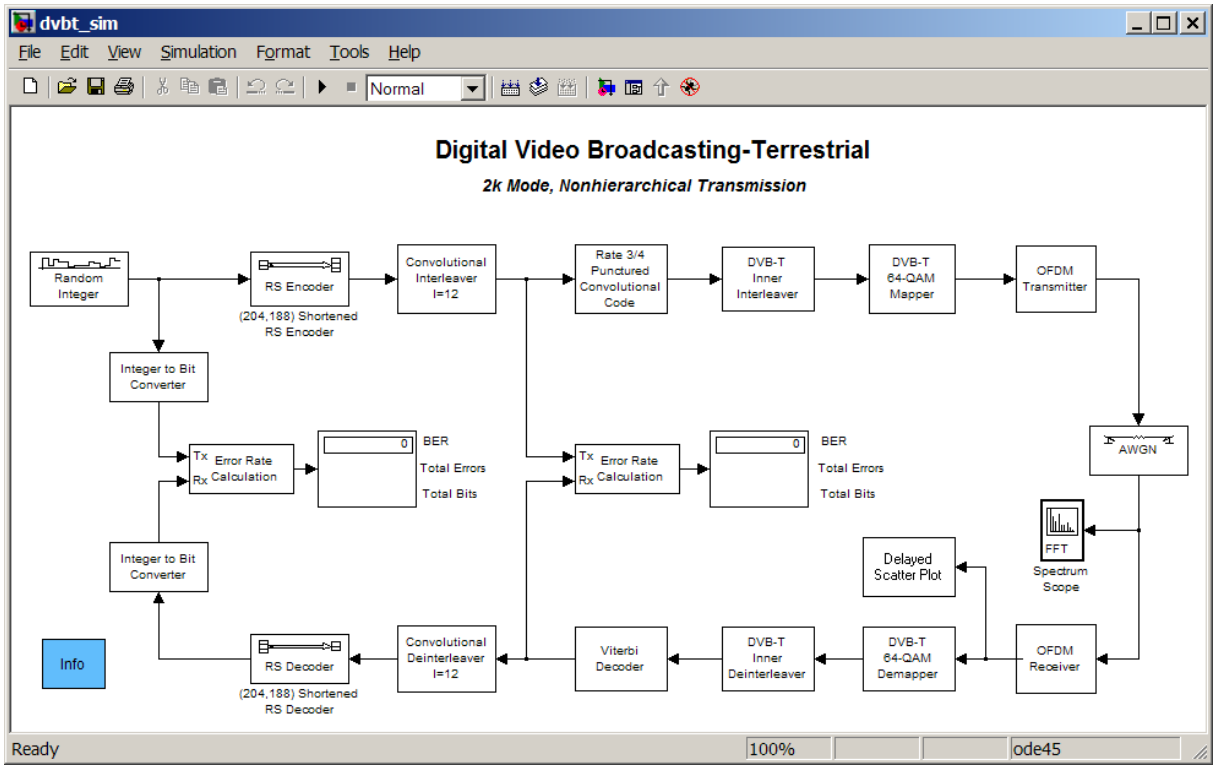
• Nous allons ajouter un oscilloscope 'flottant' qui va nous permettre d'observer n'importe lequel des liens (1 lien = 1 signal) du modèle pendant la simulation.
Faites glissez le bloc *Scope* de la fenêtre de la librairie *Simulink/Sinks* sur la fenêtre du modèle *thermo*.
Dans la fenêtre *thermo*, double cliquez sur le bloc *Scope*. Une fenêtre s'ouvre. Dans cette fenêtre cliquez sur l'icône *Properties* et cochez *Floating scope*.
Lancez la simulation. Sur le modèle sélectionnez un lien ; dans la fenêtre de l'oscillo flottant cliquez sur l'icône en forme de jumelles et observez. Vous pouvez répétez cette opération sur un autre lien sans arrêter la simulation.

• Nous allons modifier des paramètres du modèle au cours de la simulation.
Lancez la simulation et double cliquez sur la sinusoïde qui simule les variations de température et modifiez l'amplitude. Observez ce qui se passe sur les oscilloscopes. Puis juste après, modifiez en double cliquant sur 50 la valeur moyenne de la température extérieure et observez !

2.4 Une grosse démonstration

Dans la fenêtre MATLAB, tapez *dvbt_sim*. Cette commande lance SIMULINK et fait apparaître la fenêtre suivante qui contient un modèle de transmission en télévision numérique terrestre.

Analysez le schéma et repérez la partie émission, réception, le canal et les éléments qui vont être visualisés.
Lancez la simulation, et observez.



Ces premières manipulations illustrent la puissance d'un tel outil. Sans compter l'aspect interface qui se découvre naturellement puisque par quelques clics intuitifs vous pouvez changer le nom des boites, coloriez les éléments du modèle ...

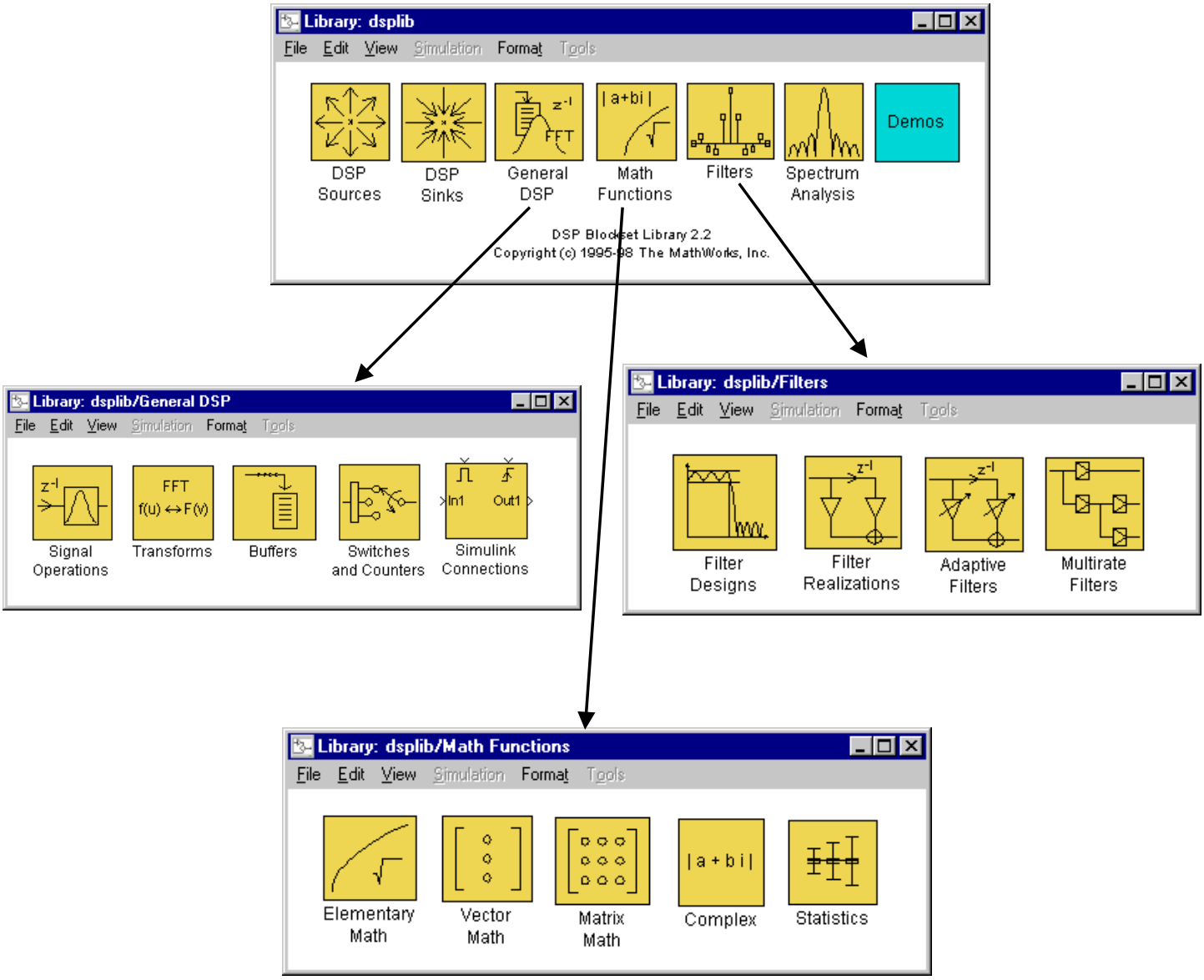
3. Le DSP Blockset

Comme MATLAB possède des Toolbox (Signal, Image, Communication, ...), SIMULINK possède des Blockset, c'est-à-dire des librairies de blocs prédéfinis et prêts à l'emploi. Nous nous intéresserons ici au DSP (Digital Signal Processing) Blockset qui regroupe des blocs équivalents aux fonctions du Toolbox Signal Processing.

3.1 Vue d'ensemble

Pour accéder à cette librairie, cliquez sur le bloc *Blocksets & Toolboxes* dans la fenêtre *Library:Simulink*, puis sur le bloc *DSP Blockset* dans la fenêtre qui apparaît.
La fenêtre *Library:dsplib* apparaît. Elle permet d'accéder aux blocs du DSP Blockset. Cette librairie est divisée en 6 sous-ensembles comme le montre le schéma suivant.

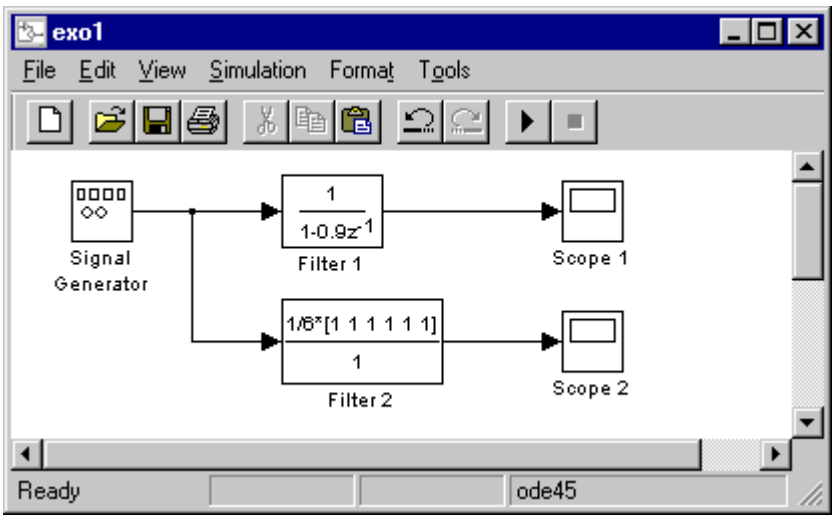
Pour accéder au contenu de chaque ensemble, il suffit de cliquer dessus. Les sous-ensembles *General DSP*, *Math Functions* et *Filters* sont eux même divisés en sous-ensembles. On pourra étudier le contenu de *General DSP*.



3.2 Construisez votre premier modèle

A partir de la fenêtre browser, ouvrez un nouveau modèle. Une Fenêtre '*untitled*' apparaît. C'est là que vous allez concevoir votre modèle et le simuler. Pensez à sauver régulièrement votre modèle par *File/Save*.

Vous allez concevoir le modèle suivant :



Pour cela, vous allez devoir aller chercher 3 blocs :

- *Simulink/Sources/Signal Generator*
- *Simulink/Sinks/Scope*
- *Simulink/Discrete/Discrete Filter*

Vous allez dupliquer les blocs *Filter* et *Scope* par un ^C ^V.
Vous allez les relier entre eux. Pour créer un lien, on utilise le bouton gauche de la souris. Pour effectuer un branchement sur un lien existant, il faut sélectionner le lien et utiliser le bouton droit de la souris.
Vous allez ensuite paramétrer les différents blocs en double cliquant dessus :

- On choisira un signal sinusoïdal d'amplitude 1 et de fréquence 2,5 Hz
- On définira les deux filtres en choisissant pour *Sample Time* : 0.01.
- En cliquant sur les scopes, on fait apparaître les fenêtres de visualisation. On explorera les rôles des différents boutons.

On va ensuite paramétrer la simulation par *Simulation/Parameters* en choisissant un *Stop Time* de 4s.

Vous pouvez lancer la simulation par *Simulation/Start* ou en cliquant sur le bouton  .