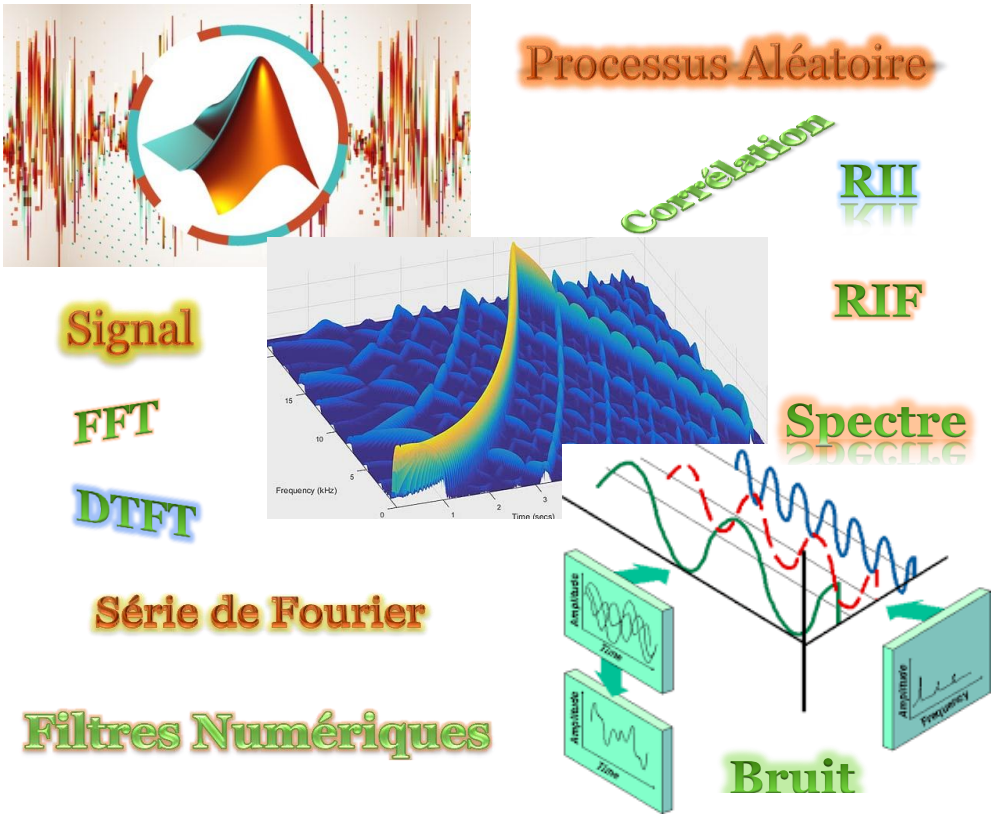
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8 Mai 1945, Guelma
Faculté des Sciences et de la Technologie
Département Electronique et Télécommunication

Travaux pratiques de Traitement numérique du signal

3éme Année Licence Electronique

5ème Semestre

Année Universitaire 2017/2018



Dr. KASSA-BAGHDOUCHE Lazhar

Table des matières

Avant-propos

Objectifs

Ce polycopié est constitué de textes de travaux pratiques (TP) élaborés pour étudier pratiquement les propriétés et théorèmes fondamentaux du traitement numérique des signaux aussi bien dans le domaine temporel qu'en domaine fréquentiel. L'accent y est mis sur l'utilisation pratique du logiciel `MatLab` pour la simulation des différents aspects du traitement numérique du signal (TNS) utile aux ingénieurs. Ces TP sont destinés aux étudiants de **troisième année licence en électronique**, automatique ainsi qu'aux étudiants de la première année master qui abordent ce domaine.

La stimulation de l'intérêt de l'étudiant pour le TNS est faite par le biais de travaux d'analyse. Dans cette perspective, les textes de travaux pratiques présentés dans ce polycopié sont constitués d'un rappel théorique suivi d'une série de manipulations numériques. Ces manipulations permettent à l'étudiant de vérifier pratiquement sur machine les différentes notions théoriques présentées dans le cours ≪ traitement du signal ≫.

Les travaux pratiques du TNS présentés dans ce polycopié se déroulent sur un semestre. A l'issu de ces TP, l'étudiant sera en mesure de :

— Grénérer et visualiser des signaux usuels, analogique et numérique, dans le domaine temporel et fréquentiel ;

— Effectuer l'interprétation géométrique des principales classes de signaux ;

— Rappeler le théorème de l'échantillonnage et choisir la fréquence d'échantillonnage ;

— Identifier et expliquer l'effet de repliement spectral qui pouvant s'écrire avec l'échantillonnage ;

— Maîtriser les séries de Fourier : représentations spectrales et calcul de la puissance ;

— Représenter et analyser le contenu fréquentiel des signaux à l'aide de la transformée de Fourier (FFT et IFFT) ;

— Utiliser les propriétés de la transformée de Fourier et la dualité temps-fréquence pour interpréter le spectre d'un signal ;

— Synthétiser et concevoir des filtres numériques à réponse impulsionnelle finie (RIF) et réponse impulsionnelle infinie (RII) ;

— Générer et visualiser les processus aléatoires (processus stochastiques) ;

— Repérer les processus aléatoires uniformes et gaussiennes en terme de leurs densité de probabilité et fonction de répartition cumulée ;

— Calculer et estimer les propriétés des processus aléatoires *discret et continue* (moyenne, variance, fonction d'autocorrélation et autocovariance).

Pré-requis

Ces travaux pratiques nécessitent la connaissance du logiciel `MatLab` et des concepts de base du traitement numérique du signal (l'autocorrélation, la convolution, la transformée de Fourier,... etc.), ainsi que des connaissances mathématiques.

Compte rendu du TP

A la fin de chaque séance (ou dans le cas échant lors de la prochaine séance de TP), il sera demandé à l'étudiant de remettre un rapport écrit contenant :

- Les fonctions et scripts `MatLab` réalisés[1] ;
- L'analyse des problèmes examinés, les opérations effectuées par `MatLab`, et les résultats obtenus.
- Le détail des calculs de prédétermination des problèmes posés ;
- Les graphiques obtenus par `MatLab`, commentés de façon à mettre en évidence la concordance (ou discordance) entre les résultats obtenus et les résultats théoriques.

Evaluation

1. Travail effectué lors des séances de TP (évaluation continue : **30%**).
2. Rapports de synthèse (**30%**).
3. Test de TP final (**40%**).

Bibliographie

[1] H.P. Hsu. *Signals and Systems*. Schaum's Outline. McGraw Hill, 1995.

[2] B.Porat. *A Course in Digital Signal Processing*. J. Wiley, 1997.

1. Chaque étudiant/binôme sera responsable de la sauvegarde de ses fichiers (Rapport écrit + programmes `MatLab`) sur la machine.

[3] D.G. Manolakis J.G. Proakis. *Digital Signal Processing*. MacMillan, 2 edition, 1992.

[4] C.S. Burrus et al. *Computer-Based Exercises for Signal Processing*. PrenticeHall, 1994.

[5] J.G. Proakis and V.K. Ingle. *Digital Signal Processing Using* `MatLab`. PWS, 1997.

[6] B.W. Jervis and E.C. Ifeachor. *Digital Signal Processing*. Addison-Wesley, 1993.

[7] M. Labarrère et al. *Le filtrage et ses applications*. Cepadues Editions, 1982.

[8] H. Leich et R. Boîte. *Les filtres numériques*. Masson, 1980.

[9] H. Lam. *Analog and Digital Filters*. Prentice Hall, 1979.

[10] C.S. Burrus and T.W. Parks. *Digital Filter Design*. J. Wiley, 1987.

[11] Ch.S. Williams. *Designing Digital Filters*. Prentice-Hall, 1986.

[12] T.W. Parks and C.S. Burrus. *DFT/FFT and convolution algorithms*. J. Wiley, 1985.

[13] R.W. Ramirez. *The FFT Fundamentals and Concepts*. Prentice-Hall, 1985.

[14] G. Blanchet and M. Charbit. *Traitement numérique du signal*. Hermès, Paris, 1998.

[15] F. de Coulon. *Théorie et traitement des signaux*. Dunod, Paris, 1984.

[16] P. Duvaut. *Traitement du signal, concepts et applications*. Hermès, Paris, 1991.

[17] A. Papoulis. *Probability, random variables and stochastic processes*. McGraw-Hill, 1984.

[18] Francis Cottet. *Aide-Mémoire de traitement du signal*. Dunod, Paris, 2011.

[19] Luc Jolivet et Rabah Labbas. *Théorie élémentaire du signal : Rappel de cours et exercices corrigés*. Hermes Science, Paris, 2005.

TP 1

Prise en main de MatLab

1.1 Objectif du TP

`MatLab` *(MATrix LABoratory)* est un logiciel de calcul plus puissant qui permet de réaliser des programmes structurés, travailler sur des matrices de grande dimension, faire des calculs sur des nombres complexes et tracer des graphes en deux et trois dimensions ($2D$ et $3D$). Ce logiciel est considéré comme un langage de programmation similaire aux langages C et Pascal. En revanche, sa particularité est qu'il s'agit d'un langage interprété (c'est-à-dire, les instructions sont exécutées immédiatement après avoir été tapées). `MatLab` présente comme avantage la disponibilité d'un nombreux algorithmes et fonctions de calcul numérique et traitement du signal, ainsi que l'existence d'une aide en ligne (commande `help`). De plus, il constitué d'un noyau de base extensible à l'aide de nombreuses boîtes à outils (toolboxes) et des boites à outils personnelles.

L'objectif de ce TP est de familiariser l'étudiant avec logiciel `MatLab` qui sera utilisé pour tous les TP de traitement de signal. Il s'agit donc de lui présenter :

— Les fonctionnalités de base et les commandes les plus utiles qui interviennent dans la simulation de différents types des signaux.

— L'objet de base manipulé par `MatLab`, à savoir les vecteurs et les matrices.

— L'architecture des scripts, fonctions, ainsi que les structures de contrôles qui peuvent être utiles dans le traitement du signal.

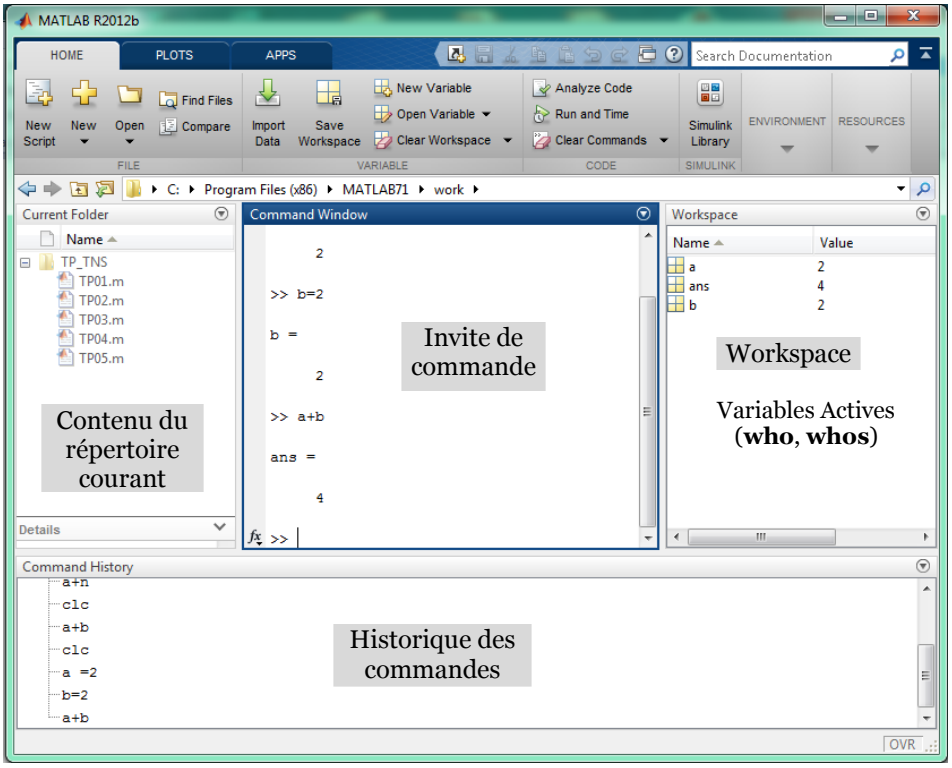
— Les commandes qui permettant de tracer les graphiques en 2D et 3D.

— La modélisation effectuée avec Simulink.

1.2 Bref descriptif de MatLab

Après avoir cliqué sur l'icône *MatLab* dans WINDOWS, vous vous trouvez dans la feuille de calcul de MatLab que l'on appellera **fenêtre** MatLab.

A partir du menu déroulant **View**, quelques fenêtres peuvent être ouvertes : les fenêtres : **Répertoire courant** *(current directory)*, **Commandes** *(command)* et **Historique** *(command history)*.



Fenêtre *Répertoire courant* : Cette fenêtre permet de sélectionner le répertoire de travail et de visualiser son contenu. Le répertoire dans lequel vous souhaitez travailler doit être sélectionné.

Fenêtre *Command Window* : Cette fenêtre permet de saisir des valeurs, de taper des instructions ligne par ligne et d'exécuter directement des programmes (en tapant le nom du programme). L'exécution des instructions et des programmes doit être validée par la touche « entrée » du clavier pour être prise en compte par MatLab.

Fenêtre *Historique* : Cette fenêtre contient toutes les instructions saisies dans la fenêtre de commandes. L'ouverture du fenêtre command history permet de voir cette liste. De plus, nous pouvons effacer le contenu de cette fenêtre en sélectionnant l'option "clear command history" dans le menu déroulant "Edit".

1.3 Les variables sous MatLab

Le logiciel MatLab gère de façons différentes les nombres entiers, réels, complexe, chaines de caractère ainsi que les tableaux de nombre.

Généralités sur les variables

Dans MatLab :

— Les variables sont définies à l'aide d'expressions ;

— Le nom de variable validé par MatLab est : lettre + nombre quelconque de lettres, chiffres ou _. Par exemple : `nom_de_variable` ;

— Aucune déclaration préalable de type de variable ou de dimension de tableau/vecteur est requiert ;

— Majuscules $\neq$ minuscules dans les noms de variables, fonctions ... etc. Par exemple : variable $abc \neq Abc$;

— Les noms de toutes les constantes et fonctions prédéfinies sont en minuscules ;

— Lorsqu'un nouveau nom de variable est déclaré, MatLab crée la variable correspondante et y associe l'espace de stockage approprié dans le *Workspace*. Dans ce cas, si la variable existe déjà, son contenu est changé et un nouvel espace de stockage est alloué en cas de redimensionnement de tableau.

Types de nombres (réels, complexes, entiers)

Réels : MatLab calcule et stocke par défaut tous les nombres en virgule flottante « *double precision* » (précision finie de 16 chiffres décimaux).

Exemple : `3, 123, -99, 0.000145, -1.6341e20, 4.521e-5.`

Entiers : les fonctions `int8`, `int16`, `int32` et `int64` génèrent des variables entières signées stockées respectivement sur `8 bits`, `16 bits`, `32 bits` ou `64 bits`.

Les opérateurs ou fonctions qui mélangeant des paramètres de types entier et réel, donnent un résultat de type entier. Alors que, dans MatLab certaines opérations qui mixent des données de type réel avec des données de type entier 64 bits ne sont pas autorisées.

Exemple : l'expression `13.3*int64(12)` génère une erreur.

Complexes : stockés sous forme de nombres réels double precision

Exemples : `nb_complexe = 4e-13 - 5.6i ou -45+5*j`.

La lettre j est utilisée pour noter le nombre complexe j ($j^2 = -1$).

On peut également extraire la partie réelle, la partie imaginaire, le complexe conjugue, le module et l'argument d'un nombre complexe (voir Annexe).

Format des nombres

Dans MatLab, les calculs sont exécutes avec une grande précision, mais les résultats ne sont affichés, de façon standard, qu'avec 4 chiffres après le point décimal. Pour afficher le résultat de façon plus précise, taper `format long`, et pour revenir au format standard, taper `format short`.

Dans l'exemple qui suit, la saisie de l'un des instructions suivie de pi donne différents valeurs de π.

`format short`	3.1416
`frmat long`	3.141592653589793
`format short e`	$3.1416e+00$
`format long e`	$3.141592653589793e+00$
`format hex`	$400921fb54442d18$

Chaînes de caractères

L'instruction `strfun` permet de voir la liste des fonctions relatives aux chaînes de caractères :

Pour définir une chaıne de caractères, on utilise les quotes `'`. Par exemple, `mot = 'abcd'` définit une chaine de caractère stockée dans une mémoire appelée mot et contenant 4 caractères.

Dans MatLab, une chaîne est un vecteur-ligne contenant autant d'éléments que de caractères.

Les informations concernant la manipulation des chaines de caractères peuvent être obtenues dans l'aide par l'index string (=chaine de caractères).

```
>> string = 'chaîne de caractères'
```

Enregistre la chaîne de caractères sur la variable string *(= vecteur-ligne).*

Si la chaîne contient une apostrophe, il faut la dédoubler.

Exemple : `section = 'Département d''électronique'`.

L'instruction `string(i:j)` : partie de string comprise entre le i-ème et le j-ème caractère.

L'instruction `string(i : end )` ou `string(i : length(string))` : fin de la chaîne à partir du caractère i.

Exemple : `section(15:23)` ou `section(15:end)` retourne la chaîne « *électronique* ».

Opérateurs arithmétiques

Pour connaitre la liste des opérateurs et caractères spéciaux sous MatLab taper la commande `helpwin ops`.

Ces operateurs arithmétiques S'appliquent à des scalaires, vecteurs ou matrices.

Opérateurs relationnels

Les opérateurs relationnels permettent de faire des tests numériques en construisant des "expressions logiques", c-à-d des expressions retournant les valeurs vrai (1) ou faux (0).

Pour des vecteurs ou matrices, ces operateurs relationnels s'appliquent à tous les éléments et retournent donc également des vecteurs ou des matrices.

Opérateurs logiques

Les opérateurs logiques ont pour arguments des expressions logiques et retournent les valeurs logiques vrai (1) ou faux (0) (voir Annexe).

Si les expressions sont des matrices, ces operateurs logiques s'appliquent à tous les éléments et retournent donc également des des matrices.

1.4 Vecteurs et Matrices

Définition

Une *matrice* est un tableau de nombres. Sur le plan mathématique d'algèbre linéaire, elle peut représenter une application linéaire.

Un *vecteur* est une matrice particulière. Du point de vue des mathématiques, les vecteurs se présentent sous formes de colonnes, mais en informatique, un vecteur est simplement une suite finie de nombres, mis sous forme de colonnes ou de lignes.

Affectation de vecteurs = matrice 1xn (ligne) ou nx1 (colonne)

Il y a deux façons simples de définir un vecteur en MatLab :

Pour créer un *vecteur ligne*, on utilise : `V = [v1 v2 ... vn]`

Exemple : `V1 = [1 -4 5], V2 = [-3,sqrt(4)], V3 = [V2 V1 -3]`

Pour créer un *vecteur colonne*, on utilise : `V = [v1 ; v2 ;... ; vn]`

ou `V = [v1 ; v2 ;... ; vn]'` (c'est-à-dire la transposée du vecteur V)

Exemple : `V4 = [-3;5;2*pi], V5 = [11 ; v4], V6 = [3 4 5 6]'`

Il est aussi possible de créer un vecteur ligne ou colonne avec l'operateur « : » :

`V = [début:pas:fin]` ;

Si le pas est non spécifié, les vecteurs sont définit avec une incrémentation unité.

Exemple : `V7 = [1 :5], V8 = [1 :2 :10]'`

Adressage de vecteurs

Pour accéder à une valeur particulière d'un vecteur, on écrira par exemple

`V(i)` : qui donne simplement accès au ième élément du vecteur ligne ou colonne `V` ;

`V(i:p:j)` : donne accès aux éléments qui vont d'indices `i` à `j` du vecteur ligne ou colonne `V` avec un pas de 1 ou de p si spécifié ;

`V([i j k:l])` : donne accès aux éléments qui vont d'indices `i`,`j` et `k` à `l` ;

`V(:)` le symbole « : » sans autres précisions signifiant tout le vecteur.

Les opérations et fonction sur les vecteurs sont résumés dans l'annexe.

Affectation de matrices

Pour `MatLab`, une matrice est un tableau à 2 dimensions de `NxM` éléments (`N` lignes et `M` colonnes).

Il est aussi possible de créer une matrice avec l'operateur « [ ] », et accéder aux ses éléments avec l'operateur «( )» (comme pour les vecteurs).

Pour crée une matrice `n` lignes x `m` colonnes, on utilise :

```
M = [v11 v12 ... v1m ;
     v21 v22 ... v2m ;
     ... ... ... ... ;
     vn1 vn2 ... vnm ] ;
```

Exemples : `M1 = [-2:0 ; 4 sqrt(9) 3].`

`M2 = [[1 2 6]' [2 4 7]' [3 5 8]'].`

`V1 = 1:3:7 et V2 = 9:-1:7, M3 = [v2;v1].`

Adressage de matrices

Pour accéder à une valeur particulière d'un matrive, on écrira par exemple :

`M(i,j)` : qui donne accès au élément `(i,j)` du matrice `M` ;

`M(i:j,k:m)` : qui donne accès au sous-matrice (lignes `i` à `j`, colonnes `k` à `m`) ;

`M(i,:)` : qui donne accès aux éléments du ligne `i` ;

`M(i:j,:)` : qui donne accès aux éléments des lignes `i` à `j` ;

`M(:,k)` : qui donne accès aux éléments de colonne `k` ;

`M(:,k:m)` : qui donne accès aux éléments des colonnes `k` à `m`.

Il existe un certain nombre de commandes (voir annexe) qui génèrent automatiquement des matrices particulières comme une matrice de taille quelconque contenant des coefficients identiques.

Il est aussi possible d'utiliser des commandes pour créer des matrices non carrées, par exemple `eye(5; 3)`. Ainsi que, des matrices par blocs. Dans ce cas, il suffit d'écrire :

```
B = [eye(3) eye(3) ;ones(3) zeros(3) ;]
```

ou bien en utilisant les colonnes

```
B = [[eye(3) ones(3)]' [eye(3) zeros(3)]']
```

ou bien encore

```
B = [[eye(3) ; ones(3)] [eye(3) ;zeros(3)]]
```

1.5 Structure de Contrôles

Dans leur forme élémentaire, les structures de contrôle de `MatLab` fonctionnent de la même manière que dans la plupart des autres langages de programmation.

La boucle `for`

Cette boucle permet d'exécuter une séquence d'instructions de manière répétée. Cette boucle consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données (borne supérieur et borne inferieur).

Syntaxe :

```
for indice = borne_inf : pas: borne_sup
    <Séquence d'instructions>;
end
```

Où :

`indice` : est une variable appelée l'indice de la boucle ;

`borne_inf` et `borne_sup` : sont deux constantes réelles (paramètres de la boucle) ;

`séquence d'instructions` : le traitement à effectuer pour les valeurs d'indices variant entre `borne_inf` et `borne_sup` avec un pas défini (corps de la boucle).

Par exemple, étant donnée un nombre n, le programme suivant calcul $n!$.

```
n = 4;
nfac = 1;
for k = 1:n
     nfac = nfac*k;
end
nfac
```

La boucle `while`

Cette boucle permet d'exécuter une séquence d'instructions de manière répétée. Dans cette boucle les instructions sont exécutées de façon répétée tant que la condition est satisfaite.

Syntaxe :

```
while expression logique
     <séquence d'instructions>;
end
```

Où :

`expression logique` est une expression dont le résultat peut être vrai ou faux ;

`séquence d'instructions` est le traitement à effectuer tant que expression logique est vraie.

Le traitement de séquence d'instructions est exécuté sous forme d'une boucle tant que l'expression logique est vraie. Dans le cas où l'expression logique devient fausse, le traitement des séquences d'instruction est terminé et le programme passe immédiatement à l'instruction qui suit l'instruction de fin de boucle `end`.

Par exemple, le programme suivant détermine la fraction du nombre n ($n!$) :

```
n = 4;
k = 1;
nfac = 1;
while k <= n
       nfac = nfac*k;
       k = k+1;
end
nfac
```

La boucle `if`

Cette boucle permet d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable.

La boucle conditionnelle la plus simple a la forme suivante :

Syntaxe :

```
If expression logique
    <séquence d'instructions>;
end
```

Où :

`expression logique` est une expression dont le résultat peut être vrai ou faux ;

`séquence d'instructions` est le traitement à effectuer si expression logique est vraie.

La séquence d'instructions est exécutée uniquement pour un résultat de l'évaluation de l'expression logique vraie (c'est-à-dire vaut 1). Dans le cas contraire, l'exécution des séquences des instructions est arrêtée et le programme passe à l'exécution de l'instruction qui suit la fin de la boucle `end`.

Contrairement à certains langages de programmation, il n'y a pas de mot clé `then` dans cette instruction conditionnée. Notez également que la marque de fin de bloc conditionné est le mot clé `end` et non pas `endif`.

Il existe une séquence conditionnée sous forme d'alternatives :

Syntaxe :

```
If expression logique
    <séquence d'instructions 1>;
else
    <séquence d'instructions 2>;
end
```

Si expression logique est vraie, la séquence d'instructions 1 est exécutée. Dans le cas contraire, (c'est-à-dire dans le cas où l'expression logique est faux.) c'est la séquence d'instructions 2 qui est exécutée. Le déroulement du programme reprend ensuite à la première instruction suivant le mot clé `end`.

Il est possible d'effectuer un choix en cascade :

Syntaxe :

```
if expression logique 1
    <séquence d'instructions 1>;
elseif expression logique 2
    <séquence d'instructions 2>;
    ...
elseif expression logique N
    <séquence d'instructions N>;
else
    <séquence d'instructions par
défaut>;
end
```

Dans ce cas, si l'expression logique 1 est vraie la séquence d'instructions 1 est exécutée et le programme reprend ensuite à la première instruction suivant le mot `end`. Alors que si l'expression logique 2 est vraie la séquence d'instructions 2 qui est exécutée et le programme reprend ensuite à la première instruction suivant le mot `end`,... etc. Si aucune des expressions logiques 1 à N n'est vraie la séquence d'instructions par défaut qui est exécutée.

Généralement, un choix en cascade est utilisé lors d'initialisation de données. Par exemple, pour affecter à une matrice A des valeurs qui dépend de leur indice, on initialise la matrice de la manière suivante :

```
nrows = 10;
ncols = 10;
A = ones(nrows, ncols);
for r = 1:nrows
    for c = 1:ncols
        if r == c
           A(r,c) = 2;
        elseif abs(r - c) == 1
            A(r,c) = -1;
         else
            A(r,c) = 0;
         end
    end
end
```

La boucle `switch`

La boucle `switch` est une alternative de l'utilisation d'une séquence d'instructions conditionnées pour effectuer un choix en cascade existe.

Syntaxe :

```
switch var
case cst1,
   <séquence d'instructions 1>;
case cst2,
    <séquence d'instructions 2>;
    ...
case cstN,
    <séquence d'instructions N>;
otherwise
    <séquence d'instructions par défaut>;
end
```

Où :

`var` est une variable numérique ou chaîne de caractères ;

`cst1, ..., cstN` sont des constantes numérique ou chaîne de caractères ;

`séquence d'instructions i` : est la séquence d'instructions à exécuter si le contenu de la variable `var` est égal à la constante `csti` (`var == csti`).

Dans le cas de l'égalité de la variable `var` à l'une des constantes `cst1, ... cstN`, (par exemple `cst3`), la séquence d'instructions correspondante (ici séquence d'instructions 3) est exécutée. Le programme reprend ensuite à la première instruction qui suit la fin de la boucle `end`. Dans le cas contraire, c'est-à-dire la variable `var` n'est égale à aucune des constantes, la séquence d'instructions par défaut est exécutée.

Par exemple, le programme suivant affiche un texte diffèrent selon la valeur saisie par l'utilisateur :

```
mynumber = input('Enter a number:');
switch mynumber
case -1
  disp('negative one');
case 0
  disp('zero');
case 1
  disp('positive one');
otherwise
  disp('other value');
end
```

Interruption d'une boucle de contrôle

L'instruction `break` permet de sortir des boucles `for` et `while`. L'exécution de cette instruction se poursuit séquentiellement à partir de l'instruction suivant la fin de la boucle `end`.

L'instruction `return` provoque un retour au programme principal, et donc toutes les instructions qui suivant cette instruction ne sont pas exécutées. Généralement, elle est utilisée conjointement avec une structure de contrôle par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.

L'instruction `error(' message d''erreur ')` permet d'arrêter un programme et d'afficher un message d'erreur.

L'instruction `warning(' message de mise en garde ')` permet d'afficher un message de mise en garde sans suspendre l'exécution du programme. Pour indiquer au `MatLab` de ne pas

afficher les messages de mise en garde d'un programme en tapant `warning off` dans la fenêtre de commandes.

L'instruction `pause` permet d'interrompre l'exécution du programme. L'exécution normale reprend dès que l'utilisateur enfonce une touche du clavier. L'instruction `pause(n)` suspend l'exécution du programme pendant n secondes.

1.6 Scripts et fonctions

Les fichiers M-files qui ont une extension **.m* contient des instructions MatLab. Ces fichiers peuvent être édités avec tout éditeur de texte. Dans MatLab, on distingue deux types de M-files : les *scripts* (ou programmes MatLab) et les *fonctions*.

Les scripts sont des suites d'instructions enregistrées dans un fichier appelé fichier M d'extension *(*.m)*. Ces fichiers n'ont pas d'arguments d'entrée/sortie et travaillent dans le Workspace. Ils sont créés avec le logiciel MatLab en sélectionnant *Nouveau fichier* dans le menu déroulant *Fichier*.

Toutes les variables créées/modifiées lors de l'exécution d'un script sont des variables globales qui peuvent également être déclarées par global, et donc sont des variables visibles dans le Workspace.

Les fichiers scripts sont composés de trois blocs essentiels :

1. Bloc d'acquisition des données (déclarer les variables).
2. Bloc de traitement :
   - Programmer des opérations répétitives ;
   - Effectuer les opérations algébriques ;
   - Appeler les fonctions ;
3. Bloc d'affichage (tracer les courbes).

Les fonctions, quant à elles, sont des fichiers **.m* qui permet d'enregistrer une succession d'opérations en spécifiant des arguments d'entrée/sortie. Les fonctions n'interagissent avec le workspace qu'à travers leurs variables d'entrée/sortie, les autres variables manipulées restant internes (locales) à ces fonctions. Ces variables ne sont pas disponibles à l'invite MatLab.

Syntaxe : `function [a, b] = ma_fonction (x,y)`

L'instruction `function` permet donc de construire une nouvelle fonction. La fonction se développe dans un fichier différant. En revanche, à partir *d'arguments d'entrée*, elle détermine des *variables de sortie*.

Par exemple, la fonction $y = \dfrac{x^2}{(2+\sin(x))}$ définie sur R est programée. Dans ce cas, x est la variable d'entrée et y est la variable de sortie.

```
function y = mafonction(x);
y = x.^2./(2+sin(x));
```

Le fichier doit être sauvegardé et nommer de la même manière que la fonction. La fonction `mafonction.m` peut être ensuite appelée par un programme ou directement dans la ligne de commande. Par exemple, `mafonction(2)` donnera 1.3749.

Dans la fonction `mafonction.m`, le point *'.'* placé avant les opérateurs permet d'appliquer directement la fonction à un vecteur ou à une matrice.

Par exemple :

```
x = -50:0.2:50;
y = mafonction(x);
figure(1)
plot(x,y)
```

Toutes les variables utilisées dans la fonction `mafonction.m` sont locales, sauf les variables en argument et les variables de sortie. Par exemple, dans le programme suivant, la variable a est locale :

```
function y = mafonction(x);
a =1;
y = x.^2./(2+sin(x))+a;
```

```
function y = mafonction(x);
global a
y = x.^2./(2+sin(x))+a;
```

1.7 Représentation graphique sous MatLab

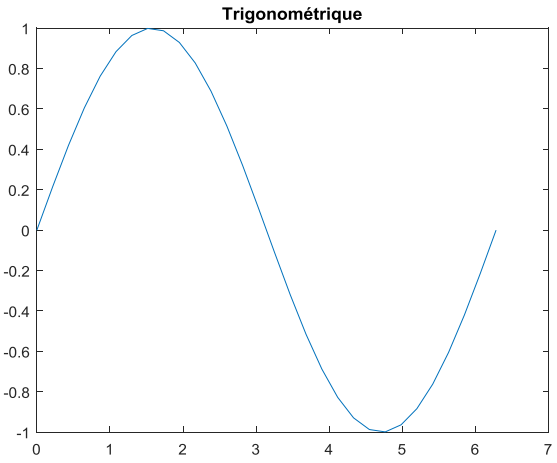
Les fonctionnalités graphiques de `MatLab` se séparent en trois catégories principales : les représentations $2D$ regroupées sous la terminologie `graph2d`, les représentations $3D$ regroupées sous la terminologie `graph3d` et les représentations spécialisées (maillage par exemple) sous la terminologie `specgraph`.

Graphiques à 2D

L'instruction `graph2d` permet de connaitre la liste des graphiques $2D$ et tableaux des fonctions disponibles.

Le programme suivant illustre un exemple :

```
x = linspace(0,2*pi,30);
y = sin(x);
figure(1);
plot(x,y, '-');
title('Trigonométrique');
```

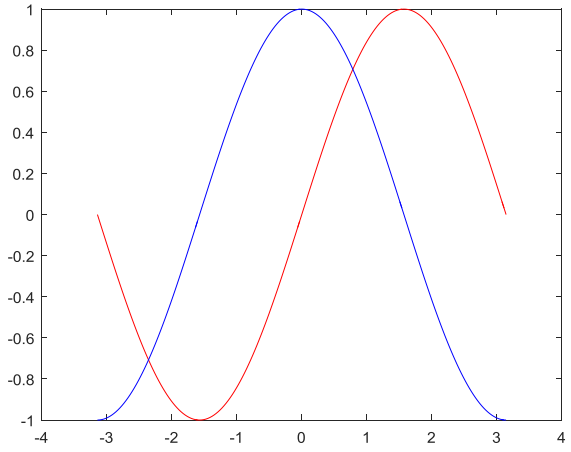


De nombreuses options existent sous `MatLab` pour **contrôler l'affichage** des courbes (Axe, couleur, commentaires, légende, échelle logarithmique,... etc) (voir Annexe).

Pour **ajouter une courbe** à un graphique existant, il suffit d'utiliser l'instruction `hold on`. Dans ce cas, les instructions graphiques qui suivent seront réalisées sur la même figure.

Par exemple, le programme suivant permet de tracer graphiquement en mode superposé les fonctions $\sin(x)$ et $\cos(x)$ sur l'intervalle $[-\pi, \pi]$ avec 201 points.

```
x = -pi:pi/100:pi;
y = sin(x);
z = cos(x);
plot(x,y, 'r');
hold on;
plot(x,z, 'b');
hold off;
```



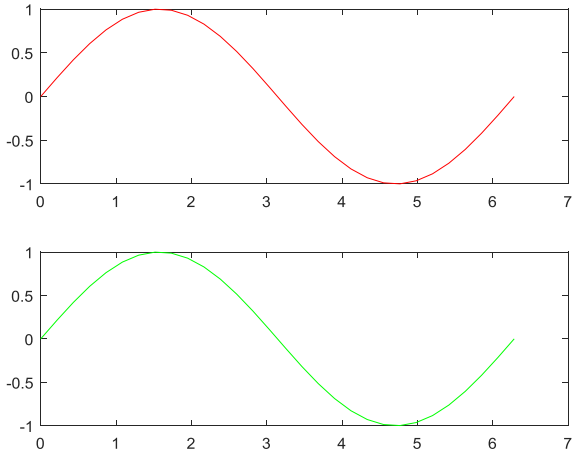
Pour **subdiviser la fenêtre graphique** et donc tracer plusieurs graphiques sur la même figure, nous utilisons l'instruction `subplot` (sous-graphique). Sa syntaxe est :

`suplot(nombre_lignes, nombre_colonnes, numéro_subdivision)`

Les subdivisions sont numéroter de 1 à $nombre\_lignes * nombre\_colonnes$, de la gauche vers la droite puis de haut en bas.

Un exemple est donné ci-dessous pour illustrer le fonctionnement de cette instruction :

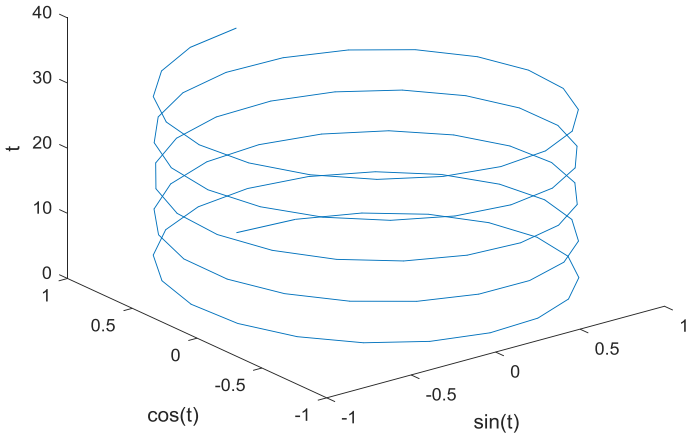
```
x = linspace(0,2*pi,30);
y = sin(x);
z = cos(x);
subplot(2,1,1);
plot(x,y, 'r');
subplot(2,1,2);
plot(x,y, 'g');
```



Graphiques à 3D

Pour tracer une courbe paramétrée avec 3 coordonnées, nous utilisons l'instruction `plot3`.

```
t = linspace(0,10*pi);
x = sin(t);
y = cos(t);
plot3(x,y,t);
xlabel('sin(t)');
ylabel( 'cos(t)');
zlabel('t');
```



Les images : image et imagesc

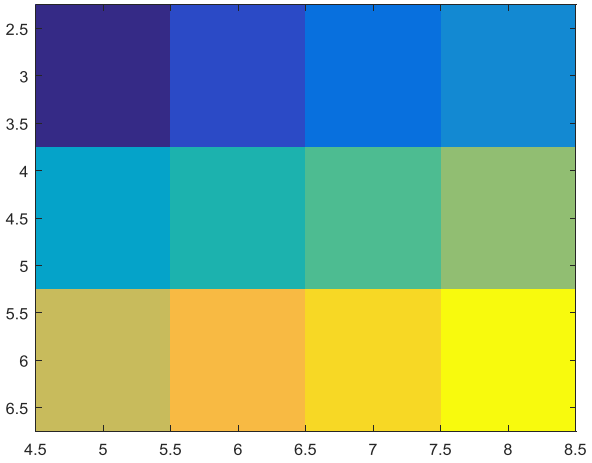
Dans Mtalab, une image pixélisée peut être vue de manière schématique comme une matrice dans laquelle chaque élément à une valeur correspondant à une couleur donnée.

L'instruction `image(x, y, A)` permet de représenter l'image contenue dans une matrice A. Celui-ci permet de spécifier les échelles des axes en prenant les valeurs contenues dans les vecteurs x et y.

L'instruction `imagesc(x, y, A)` permet d'utiliser toutes la palette de couleur disponible sans modifier le contenu de la matrice. Les lettres *sc* ajoutées correspondent à l'abréviation de *scaling*, qui signifie *renormaliser*.

Le script suivant illustre l'utilisation de l'instruction `imagesc` :

```
x = [5 8];
y = [3 6];
A = [0  2  4  6;
     8  10 12 14;
     16 18 20 22];
imagesc(x,y,A);
```

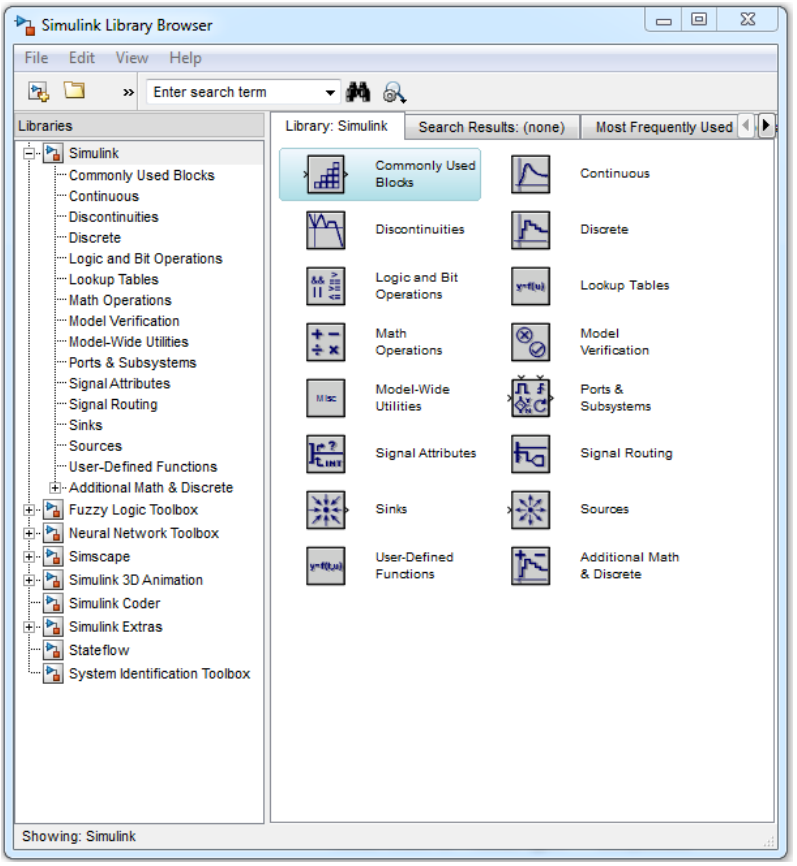


1.8 Introduction à Simulink

Simulink est l'interface graphique de `MatLab` qui permet de représenter les fonctions mathématiques et les systèmes sous forme de diagramme en blocs, et de simuler leurs fonctionnements. Simulink possède des bibliothèques de blocs, regroupés dans des *Blocksets* (Simulink Communications blockset pour les systèmes de communications par exemple). Simulink propose une approche causale de la modélisation. Le comportement dynamique d'un système est caractérisé par un bloc contenant, par exemple, la fonction de transfert du système avec une entrée et une sortie. L'information circulant dans les connexions entre deux blocs est un signal numérique orienté.

Pour démarrer Simulink, cliquer sur l'icône *Simulink bloc Library* (ou taper *Simulink* dans la fenêtre commande de `MatLab`).

Cette fenêtre contient des collections de blocs que l'on peut ouvrir en double-cliquant dessus :



`Sources`	Blocs pour générer ou importer des données de signal tels que le signal sinus
`Continuous`	Blocs des fonctions continues tels que les dérivés et l'intégrateur
`Discontinuities`	Blocs des fonctions discontinues tels que les saturations
`Discrete`	Blocs des fonctions à temps discrets tels que délais d'unité
`Logic and Bit Operations`	Blocs des opérateurs logiques ou de bits tels que l'opérateur logique et l'opérateur relationnel
`Math Operations`	Blocs des fonctions mathématiques comme le gain, le produit et la somme

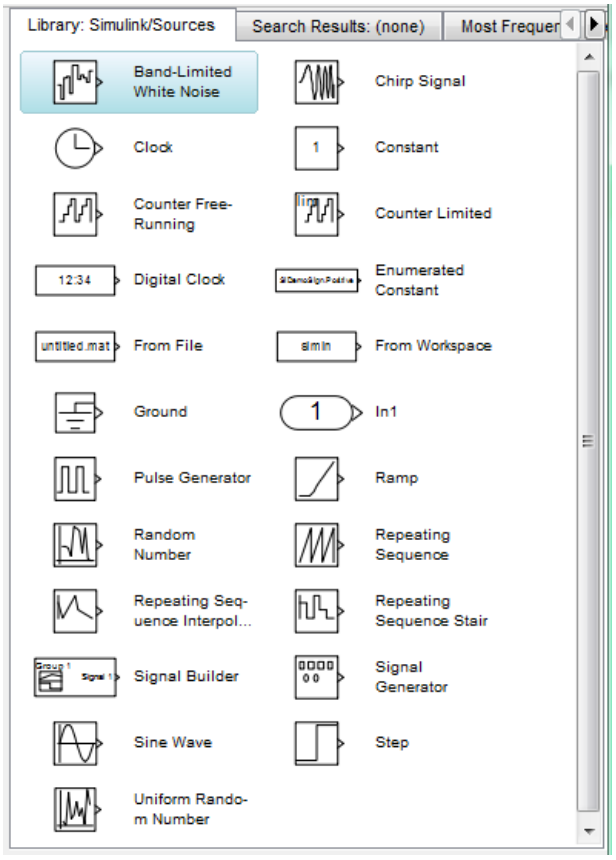
Construction d'un modèle Simulink

La création d'un nouvelle modèle sous Simulink se déroule, en général, selon les étapes suivantes :

— Choisir *New-Model* dans le menu *File*. Une fenêtre de travail nommée Untiled s'ouvrira.

— Ouvrir les collections des de blocs en double-cliquant dessus.

— Faire glisser les blocs nécessaires pour dans la fenêtre de travail.



— Faires des liaisons entres les blocs à l'aide de la souris.

— Changer les paramètres des blocs en double-cliquant sue l'icône de *bloc*.

— Fermer la fenêtre de dialogue.

— Une fois le diagramme est terminé, enregistrer le modèle dans un fichier : dans le menu *File*, choisis *Save As* et donner un nom *(*.mdl)* au fichier.

Avant de lancer la simulation, nous devons choisir les paramètres adaptés au modèle du système. Pour cela, dans le menu *Simulation*, lorsque nous choisissons Parameters, une fenêtre *Simulation Parameters* s'ouvrira. Nous devons alors choisir les paramètres pour *Solver*, *Diagnostics*, ...etc.

Pour lancer la simulation, taper sur le bouton *Start* dans le menu de la simulation.

TP 2

Génération et affichage de signaux

2.1 Objectif du TP

Le traitement du signal consiste à : étudier et analyser le signal, en extraire les informations pertinentes (ex : signaux radars) ; modifier le signal (ex : enlever les parasites d'un son ou éclaircir une image) et reproduire des signaux nouveaux (ex : voix artificielles). La plupart de ces signaux sont analogiques par nature, c'est-à-dire qu'ils sont fonction d'une variable continue, le temps, et qu'eux-mêmes varient de manière continue. Ces signaux peuvent être aussi traité et analysé sous forme numérique, à l'aide de convertisseurs analogiques-numériques (échantillonnage, quantification et numérisation des signaux analogiques).

L'objectif de ce TP est d'apprendre à **générer, visualiser et analyser** sous `MatLab` quelques signaux analogiques et échantillonnées. Ainsi que, de mettre en application les connaissances acquises sur **l'échantillonnage**, notamment le phénomène de recouvrement « *Aliasing* ».

2.2 Rappels

2.2.1 Définition

Un signal est la représentation physique d'une information à transmettre. C'est une expression d'un phénomène qui peut être mesurable par un appareil de mesure. Par exemple : les ondes acoustiques (courant délivré par un microphone (parole, musique, ...), les tensions aux bornes d'un condensateur en charge, les signaux géophysiques (vibrations sismiques), ... etc.

Le traitement du signal est un ensemble de techniques permettant de créer, d'analyser, de transformer les signaux en vue de leur exploitation. Ainsi que, d'extraction du maximum d'information utile d'un signal perturbé par le bruit.

Les principales fonctions du traitement de signal sont :

- **Synthèse :** création de signaux par combinaison de signaux élémentaires.
- **Modulation :** adaptation du signal au canal de transmission.
- **Détection :** isoler les composantes utiles d'un signal complexe, extraction du signal d'un bruit de fond.
- **Identification :** classement du signal (reconnaissance de la parole,...etc.).
- **Filtrage :** élimination de certaines composantes. Par exemple : annulation d'écho, détection de bruit sur une image,. . . etc.
- **Codage/compression** (image Jpeg, mp3, mpeg4, etc.)

2.2.2 Classification des signaux

Classification dimensionnelle

On peut classer les signaux suivant leurs dimensions :

— Signaux 1D : $x(t)$ est un signal qui évoluer en fonction d'un seul paramètre, pas forcément le temps. Par exemple : signal reçu par un microphone monophonique.

— Signaux 2D : $I(x, y)$. Par exemple : image, signal reçu par un microphone stéréophonique.

— Signaux 3D : $I(x, y, z)$. Par exemple : séquence d'images 2D.

— Signaux 4D ou 3D + t : Par exemple : séquence d'images 3D.

— Signaux N-D : des signaux sont typiquement reçus par un réseau de capteurs. Par exemple : réseaux d'antennes, ou de microphones.

Classification phénoménologique

On peut aussi classer les signaux suivant leurs caractères. Il peut être à caractère *certain* (ou déterministe) ou *aléatoire* (ou stochastique).

Signal certain : est un signal dont l'évolution en fonction du temps t est décrit par un modèle mathématique.

Signal réel : est un signal qui représentant une grandeur physique. Son modèle mathématique est une fonction réelle. Par exemple : la tension aux bornes d'un condensateur.

Signal aléatoire : est un signal dont l'évolution temporelle est imprévisible et dont on ne peut pas prédire la valeur à un temps t. La description de ces signaux est basée sur les propriétés statistiques des signaux (moyenne, variance, loi de probabilité, ...)

Classification énergétique

Dans cette classification, nous distinguons les signaux qui satisfaisant la condition d'énergie finie à ceux qui présentant une puissance moyenne finie et une énergie infinie.

On appellera énergie d'un signal $x(t)$ la quantité :

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 dt$$

et puissance moyenne de $x(t)$ la quantité :

$$P = \lim_{T_o \to \infty} \int_{-T_o/2}^{+T_o/2} |x(t)|^2 dt$$

Les *signaux à énergie finie* ont une puissance moyenne nulle. Généralement, ces signaux représentant une grandeur physique.

Les *signaux à énergie infinie* ont une puissance moyenne non nulle. Comme le cas des signaux périodiques

Classification morphologique

Dans cette classification, nous distinguons quatre types de signaux suivant la variable t (continu ou discret) :

- *Signal analogique :* le signal à amplitude et temps continus.
- *Signal quantifié :* le signal à amplitude discret et temps continu.
- *Signal échantillonné :* le signal à amplitude continue et temps discret.
- *Signal numérique :* le signal à amplitude discret et temps discret.

Classification spectrale

Les signaux sont aussi classer suivant leur répartition énergétique en fonction de la fréquence :

- Signaux de basses fréquences.
- Signaux de hautes fréquences.
- Signaux à bande étroite.
- Signaux à large bande.

2.2.3 Autocorrélation et intercorrélation

Fonction d'autocorrélation

L'autocorrélation réalise une comparaison entre un signal $x(t)$ et ses copies retardées :

Pour les signaux à énergie finie : $C_{xx}(\tau) = \int_{-\infty}^{\infty} x(t) \cdot x^*(t-\tau)dt$

$\Rightarrow$ Homogène à une énergie, $C_{xx}(0)$ est l'énergie du signal.

Pour les signaux à énergie infinie : $C_{xx}(\tau) = \lim_{T\to\infty} \int_{-T/2}^{+T/2} x(t) \cdot x^*(t-\tau)dt$

$\Rightarrow$ Homogène à une puissance, $C_{xx}(0)$ est la puissance moyenne du signal

Propriétés

1. $C_{xx}(\tau) \leq C_{xx}(0)$: Maximum en 0.
2. Si $x(t)$ est périodique alors $C_{xx}(t)$ est périodique de même période.
3. Pour des signaux réels $C_{xx}(t)$ est paire.

Fonction d'intercorrélation

L'intercorrélation, noté $C_{xy}(\tau)$, compare deux signaux $x(t)$ et $y(t)$ retardée. Cette fonction permet donc de mesurer la similitude entre ces deux signaux.

Pour les signaux à énergie finie : $C_{xy}(\tau) = \int_{-\infty}^{\infty} x(t) \cdot y^*(t-\tau)dt$

Pour les signaux à énergie infinie : $C_{xy}(\tau) = \lim_{T\to\infty} \int_{-T/2}^{+T/2} x(t) \cdot y^*(t-\tau)dt$

2.2.4 Quelques signaux élémentaires

Impulsion de Dirac

L'impulsion de Dirac $\delta(t)$, aussi appelée impulsion unité ou distribution delta, est définie par le produit scalaire :

$$x(t_o) = \int_{-\infty}^{\infty} x(t)\delta(t-t_o)dt$$

en particulier, en posant $x(t) = 1$, on obtient : $\int_{-\infty}^{\infty} \delta(t)dt = 1$

Signal signe

$$sgn(t) = \begin{cases} -1 & \text{Si } t < 0 \\ +1 & \text{Si } t > 0 \end{cases}$$

Signal saut unité (ou Echelon)

$$\Gamma(t) = 1/2 + 1/2 sgn(t) = \begin{cases} 0 & \text{Si } t < 0 \\ 1 & \text{Si } t > 0 \end{cases}$$

Signal rampe

$$r(t) = \int_{-\infty}^{t} \Gamma(\tau) d\tau = t\Gamma(t) \quad \Rightarrow \quad \Gamma(t) = \partial r(t)/\partial t \quad \text{pour : } t \neq 0$$

Signal porte ou rectangle

$$rect(t) = \Gamma(t + 1/2) - \Gamma(t - 1/2) = \begin{cases} 1 & \text{Si } |t| < 1/2 \\ 0 & \text{Si } |t| > 1/2 \end{cases}$$

Signal triangulaire

$$tri(t) = = \begin{cases} 1 - |t| & \text{Si } |t| \leq 1 \\ 0 & \text{Si } |t| \geq 1 \end{cases}$$

Exponentielle décroissante

$$y(t) = \Gamma(t) \cdot e^{-at} \quad \text{Si } a > 0$$

2.2.5 Théorème de l'échantillonnage

L'échantillonnage est une étape importante dans l'analyse du signal. Cette opération consiste à convertir un signal continu en un signal discret (c'est-à dire prélever un échantillon du signal à temps continu tous les instants).

Pour un signal continu $x(t)$, qu'on échantillonne à toutes les T_s secondes, on obtient un signal discret $x(n)$:

$$x(n) = x(t = nT_s) \quad \text{n = 0,1,2, ...,N}$$

La conversion du signal à temps continu au temps discret entraîne en général à une perte importante d'informations et à des erreurs lors de l'analyse. Pour l'éviter, le *théorème* suivant doit être respectée :

Théorème de Shannon (ou de Nyquist) : Pour un signal analogique de fréquence maximale f_{max}, la fréquence d'échantillonnage f_s doit être plus grande que $2f_{max}$. $f_s \geq 2f_{max}$.

Remarque : Le spectre d'un signal échantillonné est la somme d'une répétition périodique du spectre du signal analogique $X(f)$ en tous les multiples de la fréquence d'échantillonnage f_s.

2.3 Manipulations

2.3.1 Génération et visualisation de signaux

1. Quel est le type de signal générer par les scripts `MatLab` suivant :

Script 1

```
t=-10:1:20;
Imp=[zeros(1,10),ones(1,1),zeros(1,20)];
stem(t,Imp);
xlabel('Temps indexé en n');
ylabel('Amplitude');
```

Script 2

```
t = -1:1e-5:1 ;
x1 = rectpuls(t, 0.05);
plot(t,x1) ;
axis([-0.1 0.1 -0.2 1.2]);
xabel('Temps(sec)');
ylabel('Amplitude');
```

Script 3

```
t = -1:1e-5:1 ;
x2 = tripuls(t,0.04);
plot(t,x2);
axis([-0.1 0.1 -0.2 1.2]);
xlabel('Temps (sec)');
ylabel('Amplitude');
```

Script 4

```
fs = 10000 ;
t = 0:1/fs:1.5;
y1 = sawtooth(2*pi*50*t);
plot(t,y1);
axis([0 0.1 -1.2 1.2]);
xlabel('Temps (sec)');
ylabel('Amplitude');
```

2. Ecrire des fonctions `MatLab` permettent de générer des signaux ci-dessous décalée de t_0, d'amplitude A et de longueur arbitraire N.

(a) Une impulsion unité $Y = imp(t) = A.\delta(t - t_0)$

(b) Une fenêtre Rectangulaire $Y = A.Rec(t - t_0)$

(c) Une fenêtre Triangulaire $Y = A.Tri(t - t_0)$

3. Ecrire un script `MatLab`, faisant appel aux fonctions précédentes pour générer et représenter graphiquement chacune des séquences $x_i(n)$ suivantes :

$x_1(n) = 1.2 * \delta(n - 3)$	$-20 \leq n \leq 20$
$x_2(n) = 1.8 * Rec(t - 20)$	$N = 50$
$x_3(n) = 3.2 * Tri(t - 35)$	$N = 25$

4. Ecrire une fonction `MatLab` qui permet de générer une sinusoïde de longueur finie incluse dans l'intervalle $[t_1, t_2]$ avec les paramètres suivant : A : Amplitude, f : fréquence et Φ phase initiale.

5. Ecrire un script `MatLab`, faisant appel à la fonction précédente pour générer et représenter graphiquement chacune des signaux suivants :

$x_1(t) = 4\cos(4\pi t/4)$	$t \leq 2s$
$x_2(t) = 3\cos(4t) + \sin(\pi t)$	$t \leq 20s$

6. Ecrire un script `MatLab` qui permet de générer le signal complexe suivant : $z(n) = e^{(jn/2)}$ pour : $0 < n < 50$.

7. Représenter graphiquement le signal $z(n)$ avec les commandes : `plot(z)`, `plot(n,z)` et `stem(z)`. Qu'observez-vous ?

2.3.2 Échantillonnage des signaux analogiques

1. Ecrire un script `MatLab` qui permet de générer et représenter un signal sinusoïdal $x(t) = A\sin(2\pi ft)$ d'amplitude $A = 1$, des fréquences maximums $f = 5\ Hz, 30\ Hz$ et $80\ Hz$ et échantillonnés à $100\ Hz$. Le vecteur de temps contient les instants des échantillons avec une période d'échantillonnage T_e, $t = (0 : N) * T_e$, (N : nombre total d'échantillons).

2. Représenter le module de la transformée de Fourier discret *TFD* du signal $x(t)$.

3. Le théorème d'échantillonnage est-il respecté ? Que concluez-vous ?

4. Ecrire un script `MatLab` qui permet de générer et représenter un signal sinusoïdal d'amplitude $A = 1$, de fréquence maximum $f = 1000\ Hz$ et échantillonnés respectivement à $20000\ Hz$, $5000\ Hz$ et $1500\ Hz$.

5. Le théorème d'échantillonnage est-il respecté ? Que concluez-vous ?

6. Ecrire un script `MatLab` qui permet de générer et représenter graphiquement un signal :

   — Carré de durée 1.5 *secondes*, de fréquence $f = 180\ Hz$ et échantillonné à une fréquence de $10\ kHz$.

— Dent de scie de durée 1.5 *secondes*, de fréquence $f = 180\ Hz$ et échantillonné à une fréquence de 10 kHz.

— Représenter le module de la transformée de Fourier discret *TFD* de chaque signal. Qu'observez-vous ?

TP 3

Séries de Fourier

3.1 Objectif du TP

La décomposition en série de Fourier (ou décomposition en une somme de sinusoïdes) permet d'obtenir le spectre du signal, formé par l'ensemble des composantes sinusoïdales, et représenté par des raies discrètes situées à des fréquences multiples de la fondamentale, appelées harmoniques. La connaissance de ce spectre permet d'analyser les systèmes de modulation et de transmission de données : évaluer la bande passante nécessaire au système lorsqu'il s'agit de transmettre le signal sans déformation.

L'objectif de ce TP est donc d'étudier les fonctions périodiques par les séries de Fourier. Cette étude comprend deux volets :

— **L'analyse**, qui consiste en la détermination de la suite de coefficients de Fourier et l'énergie totale du signal $x(t)$;

— **La synthèse**, qui permet de retrouver, en un certain sens, le signal $x(t)$ à l'aide de la suite de ses coefficients.

3.2 Rappels

3.2.1 Principe et définition

La décomposition en Série de Fourier (DSF) consiste à exprimer un signal $x(t)$ de période T comme une combinaison linéaire de fonctions sinusoïdales de fréquences multiples de $F = 1/T$ dite fréquence fondamentale.

Soit un signal $x(t)$ périodique de période T, alors ce signal s'exprime sous certaines conditions comme :

$$x(t) = a_o + \sum_{n=1}^{\infty}(a_n \cos(n\frac{2\pi}{T}t) + b_n \sin(n\frac{2\pi}{T}t))$$

$\Rightarrow$ Somme de sinus et de cosinus : **facile à interpréter**

3.2.2 Coefficients de Fourier

Soit un signal $x(t)$ définit sur un intervalle de longueur $T = \frac{2\pi}{\omega}$ de la forme $[\alpha, \alpha + T[$ Alors, les coefficients a_n et b_n s'expriment en fonction du signal $x(t)$ de la manière suivante :

$$a_o = \frac{1}{T}\int_{\alpha}^{\alpha+T} x(t)dt$$

et pour $n \geq 1$:

$$a_n = \frac{2}{T}\int_{\alpha}^{\alpha+T} x(t)\cos(n\omega t)dt$$

$$b_n = \frac{2}{T}\int_{\alpha}^{\alpha+T} x(t)\sin(n\omega t)dt$$

Réciproquement, les formules précédentes constituent ce qu'on appelle les coefficients de Fourier du signal $x(t)$ et la série trigonométrique correspondante est dite la série de Fourier de $x(t)$.

3.2.3 Propriétés des séries de Fourier

— Si le signal $x(t)$ est une *fonction paire* alors $\forall n \in \mathbb{N}$, $b_n = 0$ et

$$a_0 = \frac{2}{T}\int_{0}^{T/2} x(t)dt$$

$$a_n = \frac{4}{T}\int_{0}^{T/2} x(t)\cos(n\omega t)dt$$

— Si le signal $x(t)$ est une *fonction impaire* alors $\forall n \in \mathbb{N}$, $a_n = 0$ et

$$b_n = \frac{4}{T}\int_{0}^{T/2} x(t)\sin(n\omega t)dt$$

Théorème de Dirichlet

Si le signal périodique $x(t)$ est dérivable par morceaux, alors sa série de Fourier converge partout et on a :

$$a_o + \sum_{n=1}^{\infty}(a_n \cos(n\frac{2\pi}{T}t) + b_n \sin(n\frac{2\pi}{T}t)) = \begin{cases} x(t) & \text{Si } x(t) \text{ est continu en } t \\ \frac{1}{2}[x(t_+ + x(t_-)] & \text{Sinon} \end{cases}$$

Où $x(t_+)$ et $x(t_-)$ désignent respectivement les limites à droite et à gauche de du signal $x(t)$ au point t.

3.2.4 Energie et formule de Parseval d'un signal

Energie

L'énergie d'un signal $x(t)$ de période T sur l'intervalle $[\alpha, \alpha + T[$, est la quantité :

$$E(x) = \frac{1}{T}\int_{\alpha}^{\alpha+T} |x(t)|^2 dt$$

L'énergie d'une harmonique de rang $p \geq 1$ sur un intervalle de longueur T, se définie de la manière suivante :

$$E_p = \frac{1}{T}\int_{\alpha}^{\alpha+T} |a_p \cos(p\omega t) + b_p \sin(p\omega t)|^2 dt = \frac{a_p^2 + b_p^2}{2}$$

Formule de Parseval

La somme de toutes les énergies des harmoniques de la série de Fourier d'un signal $x(t)$ périodique et convergente donne une formule dite de Parseval définit comme suit :

$$\frac{1}{T}\int_{\alpha}^{\alpha+T} |x(t)|^2 dt = a_0^2 + \sum_{p=1}^{\infty} \frac{a_p^2 + b_p^2}{2}$$

3.2.5 Coefficients complexes de Fourier

L'expression complexe des séries de Fourier sont obtenir grâce à l'utilisation des formules d'Euler. On retiendra donc que l'écriture complexe de la série de Fourier du signal $x(t)$ comme suit :

$$x(t) = \sum_{n=-\infty}^{\infty} (c_n \exp(in\frac{2\pi}{T}t)$$

Où : c_n, $n \in \mathbb{Z}$ sont appelés les coefficients de Fourier du signal $x(t)$:

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} x(t) \exp(-in\frac{2\pi}{T}t) dt$$

Ces coefficients forment la représentation fréquentielle de $x(t)$.

On dira qu'elle est convergente si et seulement si la série $\sum_{n\geq 0}(c_n \exp(in\frac{2\pi}{T}t) + c_{-n}\exp(-in\frac{2\pi}{T}t))$ est convergente.

3.2.6 Propriétés des coefficients complexes de Fourier

- Si le signal $x(t)$ est réel, $c_{-n} = c_n^*$: les coefficients sont nécessairement complexes conjugués pour restituer x réel car est complexe
- Si le signal $x(t)$ est réel et pair, $c_{-n} = c_n \Rightarrow b_n = 0$
- Si le signal $x(t)$ est réel et impair,$c_{-n} = -c_n \Rightarrow a_n = 0$
- *Théorème de Parseval* : la puissance du signal périodique est : $p = \sum_{n=-\infty}^{\infty} |c_n|^2$
- Les coefficients cn sont complexes en général : $c_n = |c_n| \exp(iarg(c_n))$

3.3 Manipulation

3.3.1 Série de Fourier et l'énergie totale d'un signal créneau

Soit $x(t)$ un signal créneau pair, périodique de période 1 défini comme suit :

$$x(t) = \begin{cases} 1 & \text{Si } \ 0 \leq t \leq 1/4 \\ 0 & \text{Si } \ 1/4 \leq t \leq 1/2 \end{cases}$$

1. Ecrire une fonction `MatLab` qui permet de générer et représenter graphiquement le signal $x(t)$ sur l'intervalle $[-3/2 \ \ 3/2]$.
2. Justifier que le signal $x(t)$ satisfait aux conditions de Dirichlet.
3. Ecrire un script `MatLab` qui permet de calculer l'énergie totale du signal $x(t)$.
4. Ecrire un script `MatLab` qui permet de calculer la série de Fourier de $x(t)$.
5. Calculer et représenter la somme partielle des huit premières harmoniques du signal $x(t)$. Qu'observez-vous ?
6. Calculer l'énergie de cette somme partielle, et comparer avec l'énergie totale. Que concluez-vous ?

3.3.2 Série de Fourier d'un signal complexe

Soit le signal $x(t)$ périodique défini comme suit : $x(t) = e^{-t}$

1. Ecrire un script `MatLab` qui permet de générer et visualiser le signal $x(t)$ sur l'intervalle $[-1 \;\; 3[$. Vous pouvez utiliser la périodicité pour k entier et $t \in [k, k+1[$, $x(t) = x(t-k) = e^{-(t-k)}$
2. Calculer les coefficients de Fourier complexes c_n. Vérifier le résultat obtenu avec `MatLab` ?
3. Donner la série de Fourier du signal $x(t)$.
4. Ecrire un script `MatLab` qui permet de calculer et visualiser sur un deuxième figure les trois premières harmoniques du signal $x(t)$.
5. Calculer et représenter sur le même graphe de la figure 2 la somme partielle de ces trois harmoniques $S_3(t)$.
6. Les graphes du signal $x(t)$ et de $S_3(t)$ illustrent le résultat de Dirichlet ? Que concluez-vous ?

TP 4

Transformée de Fourier

4.1 Objectif du TP

Dans le TP précèdent, nous avons étudié les signaux périodiques et leur développement en séries de sinus et de cosinus, ou d'exponentielles complexes, appelées *séries de Fourier*. Un signal périodique de période $T = \frac{2\pi}{\omega}$ apparaissait donc comme une somme infinie d'harmoniques de fréquence $n\omega, n = 1, 2, ...$ De même, il possible de représenter les signaux non-périodiques par un outil analogue à une série de Fourier, **la transformée de Fourier**. Dans ce cas, la série de Fourier est remplacée par une intégrale de Fourier, par exemple un son qui n'est pas répété ou une impulsion unique de tension. L'intégrale de Fourier fait intervenir un spectre continu de fréquences.

L'objectif de ce TP est d'aborder **les notions de base de la transformer de Fourier continue et discréte**. Puis, effectuer *l'analyse spectrale* des signaux non peridiques à l'aide des outils disponibles sous `MatLab`.

4.2 Rappels

4.2.1 La transformée de Fourier continue

Définition

Soit $x(t)$ un signal non périodique. On appelle transformée de Fourier du signal $x(t)$, le signal complexe $X(f)$ de la variable réelle f défini par l'intégrale :

$$X(f) = F[x(t)] = \int_{-\infty}^{+\infty} x(t) \exp(-i2\pi ft)dt$$

- La transformée de Fourier d'un signal $x(t)$ est une intégrale dépendant du paramètre ω.
- $X(f)$ indique la *quantité de fréquence* f présente dans le signal $x(t)$ sur l'intervalle $]-\infty, +\infty[$.
- Le signal $X(f)$ définie pour tout $\omega \in \mathbb{R}$ est une fonction complexe qui admet un spectre d'amplitude $A_f = |X(f)|$ et un spectre de phase $\Phi(f) = arg|X(f)|$.

Conditions d'existence de la TF

La transformée de Fourier de $x(t)$ est notamment définie dans chacun des cas suivants :

1. Le signal $x(t)$ est nul pour $|t|$ assez grand.
2. L'intégrale $\int_{-\infty}^{+\infty} |x(t)|dt$ est convergente.
3. Le signal $x(t)$ à énergie totale finie.

Transformée de Fourier inverse

La transformée de Fourier inverse du signal $x(t)$ est défini par l'intégrale :

$$x(t) = F^{-1}[X(f)] = \int_{-\infty}^{+\infty} X(f) \exp(i2\pi ft)dt$$

On note que $X(f)$ et $x(t)$ sont deux descriptions équivalentes (temporelle ou fréquentielle) du même signal $x(t) \Longleftrightarrow X(f)$.

4.2.2 Propriétés de la transformée de Fourier continue

Continuité :

Lorsque l'intégrale $\int_{-\infty}^{+\infty} |x(t)|dt$ existe, la transformée $X(f)$ est une fonction continue sur $\mathbb{R}$ et de plus $X(f)$ tend vers zéro pour $|\omega|$ tendant vers ∞.

Linéarité :

La transformée de Fourier est linéaire, autrement dit :

$a_1x_1(t) + a_2x_2(t) \Longleftrightarrow a_1X_1(f) + a_2X_2(f)$

Parité et imparité :

Si $x(t)$ est un signal pair alors $X(f)$ est à valeurs réelles et

$$X(f) = 2\int_{0}^{+\infty} x(t) \cos(2\pi\omega t)dt$$

Si $x(t)$ est un signal impair alors $X(f)$ est à valeurs imaginaires pures et

$$X(f) = -2i \int_0^{+\infty} x(t) \sin(2\pi\omega t) dt$$

Dérivation : $\frac{\partial x(t)}{\partial t} \Longleftrightarrow j2\pi f X(f)$

Décalage temporel : $x(t - t_o) \Longleftrightarrow exp(j2\pi f t_o) X(f)$

Cette propriété montre qu'il y a déphasage fréquentiel.

L'amplitude A_f ne change pas et la phase est modifié de $-j2\pi f t_o$.

Décalage fréquentiel : $exp(j2\pi f_o t) x(t) \Longleftrightarrow X(f - f_o)$

Inversion temporelle : $x(-t) \Longleftrightarrow X(-f)$

Conjugaison complexe : $x^*(t) \Longleftrightarrow X^*(-f)$

Changement d'échelle : La contraction dans le domaine temporel ($a \geq 1$) correspond à la dilatation dans le domaine fréquentiel et inversement :

$$x(at) \Longleftrightarrow \frac{1}{a} X(\frac{f}{a})$$

4.2.3 La transformation de Fourier discrète (TFD)

Définition

La transformée de Fourier d'un signal discret $x(n)$ est donnée par l'expression suivante :

$$X_e(f) = \sum_{n=-\infty}^{+\infty} x(n) e^{(-j2\pi n f)}$$

- La TF d'un signal discret est une fonction continue de la variable continue f.
- La fonction $X(f)$ est une fonction complexe. Si $x(n)$ est réel, pour un intervalle de fréquence $f \in [0, 1/2]$: le spectre d'amplitude $|X(f)|$ est une fonction paire et le spectre de phase $\arg(X(f))$ est une fonction impaire.

Conditions d'existence de la TFD

L'existence de la TFD est liée à la convergence absolue de la série $x(n)$. Donc, la TF d'un signal discret $x(n)$ existe si le signal est absolument sommable :

$$\sum_{n=-\infty}^{+\infty} |x(n)| < \infty$$

Pour un signal échantillonné à la fréquence Fe, sa TFD ($Xe(f)$) est périodique de période Fe

$\Rightarrow$ l'information fréquentielle est contenue dans la bande $[+Fe/2, \quad Fe/2]$:

La transformation de Fourier discrète inverse

L'expression de la TFD inverse est donnée par l'intégrale suivant :

$$x(n) = \int_{-1/2}^{1/2} X(f) e^{(j2\pi nf)} df$$

4.2.4 Propriétés de la transformée de Fourier discrète (TFD)

Linéarité : $ax(n) + by(n) \Longleftrightarrow aX(f) + bY(f)$

Décalage temporel : $x(n - n_o) \Longleftrightarrow X(f)e^{-j2\pi f n_o}$

Décalage fréquentiel : $x(n)e^{j2\pi f_o n} \Longleftrightarrow X(f - f_o)$

Dérivation : $\partial x(n)/\partial n \Longleftrightarrow j2\pi f X(f)$

Relation de Parseval (conservation de l'énergie)

$$\sum_{n=-\infty}^{+\infty} |x(n)|^2 = \int_{-1/2}^{1/2} |X(f)|^2 df$$

Relations de Plancherel

$$\begin{cases} x(n) * y(n) \Longleftrightarrow X(f) \cdot Y(f) \\ x(n) \cdot y(n) \Longleftrightarrow X(f) * Y(f) = \displaystyle\int_{-1/2}^{1/2} X(u)Y(f-u) \end{cases}$$

4.3 Manipulation

4.3.1 La transformée de Fourier d'un signal triangulaire

Soit $x(t)$ le signal triangulaire défini par :

$$x(t) = \begin{cases} 1 - |t| & \text{Si } t \in [+1, +1] \\ 0 & \text{Sinon} \end{cases}$$

Ecrire un script `MatLab` commenté pour réaliser les opérations suivantes :

1. Tracer le graphe du signal $x(t)$.
2. Tracer la transformée de Fourier $X(f)$ du signal $x(t)$ calculée de manière formelle (*expression algébrique*).
3. Tracer la dérivée $x'(t) = \partial x(t)/\partial t$ (là où elle existe) calculée de manière formelle.
4. Vérifier que la dérivée $x'(t)$ s'exprime en fonction du signal $p(t)$ comme suit :

$$x'(t) = p(t + 1/2) - p(t - 1/2)$$

   Où $p(x)$ est le signal porte défini comme suit :

$$p(x) = \begin{cases} 1 & \text{Si } x \in]-1/2, 1/2[\\ 0 & \text{Sinon} \end{cases}$$

5. Tracer la transformée de Fourier de cette porte $P(f)$ calculée de manière formelle.
6. En utilisant le signal $P(f)$, donner la transformée de Fourier du signal $x'(t)$.
7. Tracer la transformée de Fourier $X'(f)$.
8. En utilisant la convolution des signaux (formule $x(t) = p(t) * p(t)$), recalculer la transformer de Fourier du signal $x(t)$. Que concluez-vous ?

4.3.2 La transformée de Fourier d'un signal complexe

Soit $x(t)$ un signal exponentiel complexe :

$$x(t) = Ae^{j\omega t}$$

Ecrire un programme `MatLab` qui permet de réaliser les opérations suivantes :

1. Générer et tracer 128 échantillons d'une somme de deux exponentiels complexes de fréquence $f_1 = 1680\ Hz$ et $f_2 = 1780\ Hz$ échantillonnées à $f_e = 8\ kHz$, et d'amplitude $A = 1$.
2. Tracer la partie réelle et imaginaire du signal $x(n)$.
3. Tracer la transformée de Fourier discret (TFD)$X(f)$ des premiers échantillons ($M = 32$) du signal $x(k)$ sur N points fréquentiels ($N = 256$ et 1024). Qu'observez-vous ?

4. Les sinusoïdes sont-elles résolues ? Interpréter les résultats obtenus en calculant $f_2 - f_1$ et $f_e/32$?
5. Refaire les calculs pour $M = 128$ échantillons et $N = 256$ points fréquentiels. Est-ce que les sinusoïdes sont résolues ?
6. Comment interpréter les résultats obtenus ?

TP 5

Analyse et synthèse de filtres numériques (RIF et RII)

5.1 Objectif du TP

Un filtre numérique est un élément qui modifie le contenu spectral du signal d'entrée en atténuant ou éliminant certaines composantes spectrales non désirées. Contrairement aux filtres analogiques, qui sont réalisés à l'aide d'un arrangement de composants électroniques (résistance, condensateur, etc.), les filtres numériques sont réalisés soit par des circuits intégrés dédiés ou par des processeurs programmables (microprocesseur, microcontrôleur, FPGA etc...). Ces filtres numériques sont classés en deux catégories : filtre à réponse impulsionnelle finie (RIF) et filtres à réponse impulsionnelle infinie (RII).

L'objectif de ce TP est d'**analyser et synthétiser des filtres numériques RIF et RII**. Nous intéressons, plus particulièrement, à la synthèse de filtres RIF par la *méthode de la fenêtre* (rectangulaire, triangulaire, de Hamming et de Kaiser), et de filtres RII par la *transformation bilinéaire*.

5.2 Rappels

5.2.1 Filtre numérique : définition et caractéristiques

Un filtre numérique est un *système Linéaire Discret (SLD)* qui modifie la représentation temporelle et fréquentielle de signaux discrets. Par exemple, pour des signaux radio ou des images ces filtres peuvent être utilisés pour réduire le bruit ou modifier certaines plages de fréquences.

Un filtre numérique peut être représenté par plusieurs types de spécifications :

1. **Fonction de transfert en z :**
   La fonction de transfert du filtre $H(z)$ est rationnelle :

$$H(z) = \frac{N(z)}{D(Z)} = \frac{\sum_{i=o}^{N} b_i \cdot z^{-i}}{1 + \sum_{i=o}^{N} a_i \cdot z^{-i}}$$

2. **Réponse impulsionnelle :** La transformée en z inverse de $H(z)$:

$$H(z) = \sum_{n=0}^{\infty} h(n) \cdot z^{-n}$$

3. **Equation aux différences :**

$$y(n) = \sum_{i=o}^{N} b_i \cdot x(n-i) - \sum_{i=o}^{N} a_i \cdot xy(n-i)$$

Remarques

— Si $a_n = 0$, le filtre s'appelle non récursif ou RIF (Réponse Impulsionnelle Finie) ;
— Si $a_n \neq 0$, le filtre s'appelle récursif ou RII (Réponse Impulsionnelle Infinie) ;
— Si les pôles de H(z) sont à l'intérieur du cercle unité, le filtre est stable ;
— Si la réponse impulsionnelle est causale $h(n) = 0, \forall n < 0$, le filtre est causal.

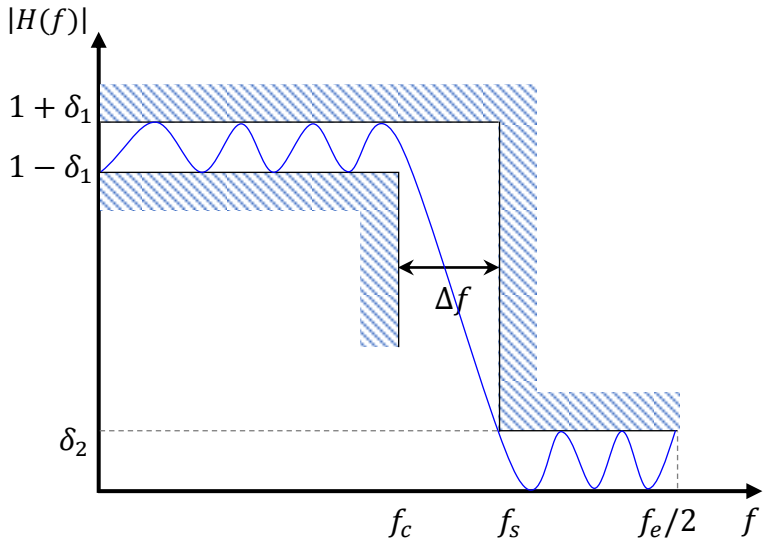
Gabarit d'un filtre numérique

Le gabarit d'un filtre numérique $H(e^{j\omega})$ est défini par l'utilisateur parmi les quatre catégories suivantes :

1. Filtre passe-bas
2. Filtre passe-haut
3. Filtre coupe-bande
4. Filtre passe-bande

Avec :

— Fe la fréquence d'échantillonnage.
— $\triangle f$ la largeur de la zone de transition.
— δ_1 l'amplitude des ondulations en bande passante.
— δ_2 l'amplitude des oscillations en bande atténuée.



5.2.2 Filtre à Réponse Impulsionnelle Finie (RIF)

Définition et caractéristiques

Les filtres RIF sont des systèmes à réponse impulsionnelle finie, de fonction de transfert $H(z)$ défini par la relation suivante :

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

dont les coefficients $h(k)$ sont tels que :

$$\begin{cases} h(k) \neq 0 & \text{pour} \quad k \in [0, N-1] \\ h(k) = 0 & \text{pour} \quad k \notin [0, N-1] \end{cases}$$

L'équation de récurrence qui liant l'entrée et la sortie montre clairement le caractère non-récursif du filtre :

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

Les *principales caractéristiques* des filtres RIF sont :

— Les filtres RIF sont toujours stables, car la sortie revient toujours à zéro après suppression de l'excitation ($\sum_{k=0}^{N-1} |h(k)| < \infty$) ;

— Les méthodes de synthèse des filtres RIF permettant de dériver n'importe quelle réponse fréquentielle ;

— Pour le même nombre de coefficients, la bande de transition des filtres RIF est toujours plus large par rapport aux filtres RII ;

— Les filtres RIF ont un temps de propagation de groupe constant et une absence de distorsion harmonique dans le signal ;

— Les filtres RIF sont plus faciles d'implémenter dans un système numérique de traitement.

5.2.3 Synthèse de filtre à RIF par la méthode de la fenêtre

Faire la synthèse d'un filtre c'est rechercher la fonction de transfert $H(z)$ correspondant à la spécification d'un gabarit imposé.

La méthode de la fenêtre consiste à appliquer une fenêtre de taille N au filtre idéal équivalent ;

Soit un filtre numérique idéal, périodique de période 2π de fonction de transfert $H(e^{j\omega})$:

$$H(e^{j\omega}) = \Sigma_{-\infty}^{+\infty} h(n) e^{-jn\omega}$$

et de réponse impulsionnelle $h(n)$

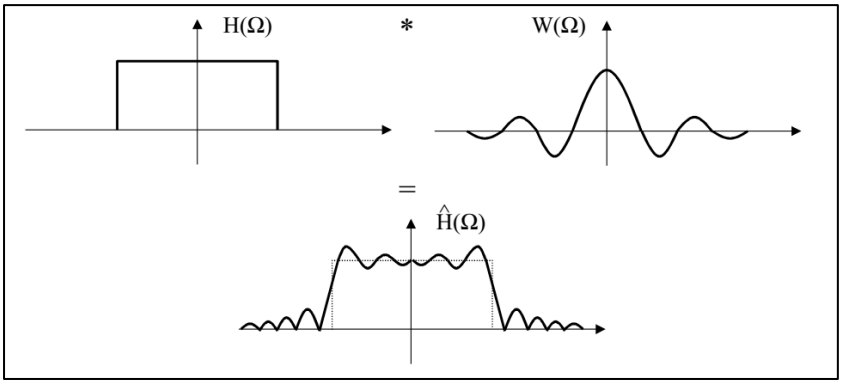
$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega}) e^{jn\omega} d\omega$$

Ce filtre est non causal et à réponse impulsionnelle infinie. La manière la plus simple d'obtenir un filtre RIF approchant $H(e^{j\omega})$ est de limiter $h(n)$ par :

$$\hat{h}(n) = \begin{cases} h(n) \cdot w(n) & \text{Si } 0 \leq n < N \\ 0 & \text{Sinon} \end{cases}$$

$$\begin{aligned} \hat{H}(e^{j\omega}) &= \sum_{n=0}^{N-1} h(n) \cdot w(n) \cdot e^{-jn\omega} \\ &= H(e^{j\omega}) * W(e^{j\omega}) \end{aligned}$$

Principales fenêtres



Rectangulaire

$$w(n) = \begin{cases} 1 & \text{Si } 0 \leq n < N \\ 0 & \text{Sinon} \end{cases}$$

Hanning

$$w(n) = \begin{cases} 0.5 - 0.5\cos(\frac{2\pi n}{N-1}) & \text{Si } 0 \leq n < N \\ 0 & \text{Sinon} \end{cases}$$

Hamming

$$w(n) = \begin{cases} 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) & \text{Si } 0 \leq n < N \\ 0 & \text{Sinon} \end{cases}$$

Blackman

$$w(n) = \begin{cases} 0.42 - 0.5\cos(\frac{2\pi n}{N-1}) + 0.08\cos(\frac{24\pi n}{N-1}) & \text{Si } 0 \leq n < N \\ 0 & \text{Sinon} \end{cases}$$

5.2.4 Filtre à Réponse Impulsionnelle Infinie (RII)

Définition et caractéristiques

Les filtres RII sont des systèmes caractérisés par des réponses impulsionnelles de durée infinie et donc les coefficients $h(k)$ sont non nuls sur l'intervalle $[0, +\infty[$. Ceci est réalisé par la présence de pôles dans la fonction de transfert du filtre :

$$H(z) = \frac{\sum_{k=0}^{p-1} b_b z^{-k}}{\sum_{i=0}^{m-1} a_i z^{-i}}$$

ce qui traduit par l'équation suivante :

$$y(n) = \sum_{k=0}^{p-1} b_k x(n-k) - \sum_{i=1}^{m-1} a_i y(n-i)$$

Les *principales caractéristiques* des filtres RII sont :

— Les filtres RII ont une instabilité potentielle due à des pôles de $H(z)$ situés en dehors du cercle unité ;

— La réponse impulsionnelle infinie permet d'obtenir un filtrage plus sélectif qu'un filtre RIF pour une quantité de calcul inférieure ;

— Les filtres RII ont une bande de transition étroite ;

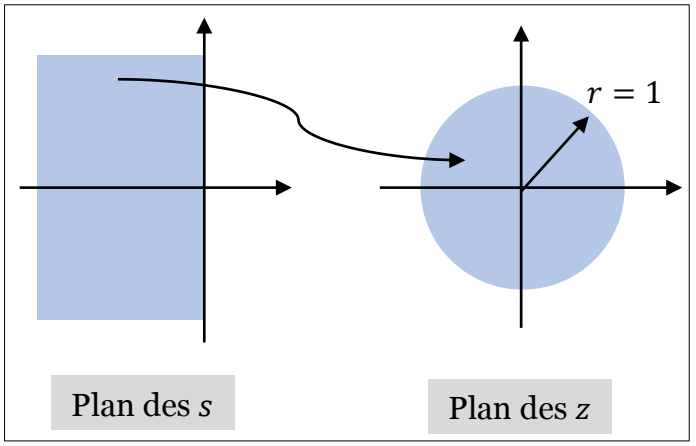
— Les filtres RII ont une complexité plus faible qu'un filtre RIF à sélectivité équivalente.

5.2.5 Synthèse de filtre à RII par la transformée bilinéaire

Une méthode pour faire la synthèse du filtre $H(z)$ à partir du gabarit numérique consiste à transformer ce gabarit en un gabarit analogique. On choisit une fonction analogique modèle et on détermine ses paramètres afin d'obtenir la fonction de transfert H (p) du filtre respectant le gabarit analogique souhaité. Enfin, on obtient la fonction de transfert H(z) du filtre numérique par application de la transformation bilinéaire :

$$H(z) = \begin{cases} H(p) \\ p = \dfrac{2}{T_e} \dfrac{1 - z^{-1}}{1 + z^{-1}} \end{cases}$$

Avec : T_e fréquence d'échantillonnage.



Les pôles de l'équation sont définit par cette équation :

$$z = \frac{2/T_e + s}{2/T_e - s}$$

Si s a une partie réelle négative, z est de module inférieur à 1 $\Rightarrow$ conservation de la stabilité.

Remarque : La transformation bilinéaire entraîne une relation non linéaire entre les fréquences analogiques f_a et les fréquences numérique f_d : $f_d = \frac{1}{\pi T_e} \tan^{-1}(\pi f_a T_e)$

5.3 Manipulation

5.3.1 Synthèse d'un filtre numérique *RIF* par la méthode de la fenêtre

Pour concevoir un filtre numérique RIF qui répond au cahier des charges (c'est-à-dire un filtre qui possédant les caractéristiques du gabarit à respecter), il est possible de calculer l'ordre N par la formule suivante :

$$N = \frac{-20log(\sqrt{\delta_1 \delta_2}) - 13}{14.6(f_s - f_p)/f_c}$$

	filtre RIF passe-haut	**filtre RIF passe-bande**
Fréquence de coupure	2400 Hz	//
Bande passante	//	1500-2500 Hz
Largeur de la zone de transition $\triangle f = f_s - f_c$	1000 Hz	500 Hz
Amplitude des ondulations en bande passante $\delta 1$	0.01 dB	0.1 dB
Amplitude des oscillations en bande atténuée$\delta 2$	$>$ 40 dB	$>$ 60 dB
Fréquence d'échantillonnage f_e	20 kHz	10 kHz

1. Écrire le script `MatLab` permettant de visualiser les fonctions de transfert des filtres RIF obtenues en utilisant les différents types de fenêtres disponibles.
2. Quels sont les avantages et les inconvénients de chacune fenêtre ?
3. Écrire le script `MatLab` permettant de visualiser la réponse impulsionnelle des filtres RIF obtenue.
4. Faire varier l'ordre du filtre dans une plage incluant l'ordre optimal N. Observer son influence sur la position de la fréquence de coupure et la réponse impulsionnelle du filtre.
5. Quelles remarques peut-on faire sur la réponse impulsionelle (*allure générale, nombre d'oscillations en fonction de l'ordre, ...*) ?

5.3.2 Synthèse d'un filtre numérique *RII* par la transformée bilinéaire

On désire réaliser un filtre numérique $H(z)$ équivalent à un filtre analogique de Chebyshev $H(j\omega)$. Le filtre numérique obéissant aux caractéristiques ci-dessous :

$$\begin{cases} \text{Fréquence de la bande passante : } f_c = 1khz \\ \text{Fréquence de la coupe bande : } f_p = 3khz \\ \text{Amplitude des ondulations en bande passante : } \delta 1 = 1dB \\ \text{Amplitude des oscillations en bande atténuée : } \delta 2 = -20dB \\ \text{Fréquence d'échantillonnage : } f_e = 10khz \end{cases}$$

1. Montrer par la méthode bilinéaire que la fonction du transfert du filtre $H(z)$ est donnée par la relation suivante :

$$H(z) = \frac{0.079(z+1)^2}{z^2 - 1.2z0.516}$$

2. Écrire le script `MatLab` permettant de visualiser la réponse fréquentielle en module et en phase du filtre numérique.
3. Trouver la relation entre l'échelle des abscisses de la commande `plot` de `MatLab` et les fréquences. Le filtre respecte-t-il le gabarit spécifié ?
4. Quelles est la relation entre la fréquence d'échantillonnage et la fréquence de coupure ?
5. Écrire le script `MatLab` permettant de visualiser la réponse impulsionnelle du filtre numérique obtenue.
6. Tracez dans le plan complexe les pôles et les zéros du filtre. Le filtre est-il stable ? Que concluez-vous ?

TP 6

Processus aléatoires

6.1 Objectif du TP

Un processus aléatoire (PA) (ou *processus stochastique*) est un processus qui décrit l'évolution d'une grandeur aléatoire en fonction du temps (ou de l'espace). Les applications principales des processus aléatoires sont : l'administration des réseaux et bien entendu dans les télécommunications. L'étude des caractéristiques de processus aléatoire on fait appel aux théories de probabilités et de statistiques dont elle constitue l'un des objectifs les plus profonds. Leur étude soulève des problèmes mathématiques intéressants et souvent très difficiles.

Ce TP aura pour objet de **familiariser les étudiants avec quelques notions de bases de processus aléatoires** : calculer les propriétés statistiques d'un ensemble de N variables aléatoire uniforme et gaussien ; calculer la fonction d'autocorrélation et la densité spectrale de puissance (DSP) des signaux aléatoire.

6.2 Rappels

Définition

Un processus aléatoire $X(t,\zeta)$ est un processus qui pour chaque temps t,$X(t,\zeta)$ est une variable aléatoire. Par ailleurs, pour $\zeta = \zeta_o$, $X(t,\zeta)$ est une fonction déterministe qui traduit une courbe appelée *trajectoire du signal.*

Exemple :

$$X(t,\zeta) = A\cos(\omega t + \theta)$$

Si θ est une variable aléatoire qui prend des valeurs entre $[0, 2\pi]$, nous pouvons avoir plusieurs réalisations. Par exemple, pour une valeur donnée θ_o de θ, $X(t, \zeta)$ est une fonction déterministe qui traduit une *trajectoire du signal aleatoire.*

6.2.1 Moyennes, moment et variance d'un processus aléatoire

Esperance mathématiques ou moyenne d'un PA

On appelle espérance mathématique ou valeur moyenne d'un PA la quantité suivante définie comme suit :

$$m_X(t) = E[X(t,\zeta)] = \begin{cases} \int_{-\infty}^{-\infty} x(t) F_1(x,t) dx & \text{Si X continue} \\ \sum_k x_k F_k(x,t) & \text{Si X discrète} \end{cases}$$

La moyenne $m_X(t)$ est définie comme moyenne au sens du calcul des probabilités.

- Si $m_X(t) = 0$, on dit que le PA est centré.
- Si tel n'est pas le cas, le signal $X(t, \zeta) - m_X(t)$ est centré.
- Si $F_1(x, t) = F_1(x)$, alors : $m_X(t) = m$ est indépendant du temps t.

Propriétés

Si X et Y sont deux VA définies sur le même univers Ω, admettant une espérance :

1. $E[X + Y] = E[X] + E[Y]$
2. $E[aX] = a.E[X]$, $\forall a \in \mathbb{R}$
3. Si $X \geq 0$, alors : $E[X] \geq 0$

Moments d'ordre n

Le moment d'ordre n d'un PA est défini comme suit :

$$E[X(t,\zeta)^n] = \begin{cases} \int_{-\infty}^{-\infty} x(t)^n F_1(x,t) dx & \text{Si X continue} \\ \sum_k x_k^n F_k(x,t) & \text{Si X discrète} \end{cases}$$

Variance d'un processus aléatoire

La variance d'un PA reel $X(t, \zeta)$ est définit comme suit :

$$\sigma_X^2(t) = Var[X] = \begin{cases} \int_{-\infty}^{-\infty} [x - E[X]]^2 F_1(x,t)dx & \text{Si X continue} \\ \sum_k [x_k - E[X]]^2 F_k(x,t) & \text{Si X discrète} \end{cases}$$

Si $\sigma_X^2(t) = 0$, le signal perd son caractère aléatoire. Il s'agit alors d'un signal déterministe.

Il est important de remarquer que l'espérance mathématique, le moment d'ordre n et la variance d'un processus aléatoire sont calculer à un instant t particulier et sont déduits uniquement à partie de la fonction de densité de probabilité $F_1(x,t)$.

6.2.2 Autocovariance, autocorrélation et stationnarité

Fonction d'autocovariance

La fonction d'auto covariance d'un processus aléatoire réel $X(t,\zeta)$, notée $C_X(t_1,t_2)$, est définie par :

$$C_X(t_1,t_2) = cov(X(t_1,\zeta), X(t_2,\zeta)) = E[X(t_1,\zeta), X(t_2,\zeta)] - m_X(t_1)m_X(t_2)$$

Dans cette définition $E[X(t_1,\zeta), X(t_2,\zeta)]$ est calculé à partir de $F_2(x_1,x_2;t_1,t_2)$.

En outre, nous remarquons que lorsque $t = t_1 = t_2$, nous retrouvons la variance du PA, c'est-à-dire : $C_X(t_1,t_2) = \sigma_X^2(t)$.

Cas des signaux complexe : Si $X(t,\zeta)$ est un PA complexe alors :

$$C_X(t_1,t_2) = E[X(t_1,\zeta), X^*(t_2,\zeta)] - m_X(t_1)m_X^*(t_2)$$

Cas des signaux discrets : Si $X(t,\zeta)$ est un PA discret alors :

$$C_X(n_1,n_2) = E[X(n_1), X(n_2)]$$

Fonction d'autocorrélation

La fonction d'autocorrélation d'un PA réel $X(t,\zeta)$, notée $\gamma_X(t_1,t_2)$, est définie par :

$$\gamma_X(t_1,t_2) = E[X(t_1,\zeta), X(t_2,\zeta)]$$

Cette fonction peut être obtenue à partir de la fonction d'autocovariance comme suit :

$$\gamma_X(t_1,t_2) = C_X(t_1,t_2) + m_X(t_1)m_X(t_2)$$

Cas des signaux complexe : Si $X(t,\zeta)$ est un PA complexe alors :

$$\gamma_X(t_1,t_2) = E[X(t_1,\zeta), X^*(t_2,\zeta)]$$

Cas des signaux discrets : Si $X(t,\zeta)$ est un PA discret alors :

$$\gamma_X(n_1,n_2) = C_X(n_1,n_2) - m_X(n_1)m_X(n_2)$$

Si $X(t,\zeta)$ est un PA centré alors : $\gamma_X(t_1,t_2) = C_X(t_1,t_2)$.

si $\gamma_X(t_1,t_2) = 0$, alors $X(t_1,\zeta)$ et $X(t_2,\zeta)$ sont orthogonaux ($X(t_1,\zeta) \perp X(t_2,\zeta)$).

Stationnarité des processus aléatoires

La notion de stationnarité est liée au cours du temps de propriétés statistiques d'un PA. Elle peut être définie de manières différentes :

Stationnaire au sens strict (SSS) : Un PA $X(t,\zeta)$ est dit SSS, si toutes ses propriétés statistiques sont invariantes dans toute translation de l'origine du temps. Ceci signifie que $X(t,\zeta)$ et $X(t+\tau,\zeta)$ ont les mêmes propriétés statistiques. C'est-à-dire :

$$\forall n \quad F_n(x_1,x_2,...,x_n;t_1,t_2,...,t_n) = F_2(x_1,x_2,...,x_n;t_1+\tau,t_2+\tau,...,t_n+\tau)$$

Stationnaire au sens large : un PA $X(t,\zeta)$ est dit stationnaire du second ordre ou stationnaire au sens large si :

$$\begin{cases} m_X(t) = m_X = constante \\ \gamma_X(t_1,t_2) = \gamma_X(t_1-t_2) \end{cases}$$

La moyenne du PA est indépendante du temps et la fonction d'autocorrélation dépend uniquement de la différence $t_1 - t_2$.

La fonction d'autocovariance dépend uniquement du retard entre les instants t_1 et t_2 : $C_X(t_1,t_2) = C_X(t_1-t_2)$

Remarque : Un SSL n'implique pas la SSS. L'inverse est également vrai

6.2.3 Intercovariance et intercorrélation

Fonction d'intercovariance

La fonction d'intercovariance de deux processus aléatoire $X(t)$ et $Y(t)$, notée $C_{XY}(t_1,t_2)$, est définit comme suit :

$$C_{XY}(t_1, t_2) = cov(X(t_1), Y(t_2)) = E[X(t_1)Y(t_2)] - m_X(t_1)m_Y(t_2)$$

Si $C_{XY}(t_1, t_2) = 0$, alors $X(t_1)$ et $Y(t_2)$ sont non corrélés. Rappelons à cet effet que deux PA non corrélés ne sont pas forcement indépendants.

En effet, soit deux PA $X(t)$ et $Y(t)$ indépendant alors :

$$\begin{aligned} C_{XY}(t_1, t_2) &= E[X(t_1)Y(t_2)] - m_X(t_1)m_Y(t_2) \\ &= E[X(t_1)]E[Y(t_2)] - m_X(t_1)m_Y(t_2) \\ &= 0 \end{aligned}$$

Cas des signaux complexe : Si $X(t)$ et $Y(t)$ sont deux PA complexe alors :

$$\begin{cases} C_{XY}(t_1, t_2) = E[X(t_1)Y^*(t_2)] - m_X(t_1)m_Y^*(t_2) \\ C_{YX}(t_1, t_2) = E[Y(t_1)X^*(t_2)] - m_Y(t_1)m_X^*(t_2) \end{cases}$$

Cas des signaux discrets : Si $X(n)$ et $Y(n)$ sont deux PA discrets, alors :

$$C_{XY}(n_1, n_2) = E[X(n_1)Y(n_2)] - m_X(n_1)m_Y(n_2)$$

Fonction d'intercorrélation

La fonction d'intercorrelation de deux processus aléatoire $X(t)$ et $Y(t)$, notée $\gamma_{XY}(t_1, t_2)$, est définit comme suit :

$$\gamma_{XY}(t_1, t_2) = E[X(t_1)Y(t_2)]$$

Cette fonction peut être obtenue à partir de la fonction d'intercovariance comme suit :

$$\gamma_{XY}(t_1, t_2) = C_{XY}(t_1, t_2) + m_X(t_1)m_Y(t_2)$$

Cas des signaux complexe : Si $X(t)$ et $Y(t)$ sont deux PA complexe alors :

$$\begin{cases} \gamma_{XY}(t_1, t_2) = E[X(t_1)Y^*(t_2)] \\ \gamma_{YX}(t_1, t_2) = E[Y(t_1)X^*(t_2)] \end{cases}$$

Cas des signaux discrets : Si $X(n)$ et $Y(n)$ sont deux PA discrets, alors :

$$\gamma_{XY}(n_1, n_2) = C_{XY}(n_1, n_2) - m_X(n_1)m_Y(n_2)$$

Orthogonalité de deux PA

Deux processus aléatoire $X(t_1)$ et $Y(t_2)$ sont dits orthogonaux $(X(t_1) \perp Y(t_2))$ si $\gamma_{XY}(t_1, t_2) = 0$

Si les deux PA sont centrés alors : $\gamma_{XY}(t_1, t_2) = C_{XY}(t_1, t_2)$

6.2.4 Matrice de corrélation et de covariance

Soit $\underline{X}(t)$ un vecteur aléatoire :

$$\underline{X}(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \\ \vdots \\ X_N(t) \end{bmatrix} \tag{6.1}$$

Moyenne du VA : la moyenne du vecteur $\underline{X}(t)$ est définie comme suit :

$$E[\underline{X}(t)] = \begin{bmatrix} E[X_1(t)] \\ E[X_2(t)] \\ \vdots \\ E[X_N(t)] \end{bmatrix} \tag{6.2}$$

Matrice de corrélation : la matrice de corrélation du vecteur $\underline{X}(t)$ est définie par :

$$\Gamma_{\underline{X}}(t_i, t_j) = E[\underline{X}(t_i)\underline{X}^H(t_i)]$$

Matrice de covariance : la matrice de covariance du vecteur $\underline{X}(t)$ est définie par :

$$C_{\underline{X}}(t_i, t_j) = \Gamma_{\underline{X}}(t_i, t_j) - = E[\underline{X}(t_i)]E[\underline{X}^H(t_j)]$$

6.2.5 Densité spectrale de puissance

La densité spectrale de puissance d'un processus aléatoire correspond à la répartition de la puissance moyenne du processus dans l'espace de fréquences. Cette densité est obtenue à partir de la transformée de Fourier de la fonction d'autocorrélation et réciproquement.

$$\begin{cases} \Phi_X(f) = \displaystyle\int_{-\infty}^{+\infty} \gamma_X(\tau) e^{-2\pi j f \tau} d\tau \\ \gamma_X(\tau) = \displaystyle\int_{-\infty}^{+\infty} \Phi_X(f) e^{2\pi j f \tau} d\tau \end{cases}$$

Ces relations sont appelées relations d'*EINSTEIN-WIENER-KINTCHINE*.

6.2.6 Ergodicité

Un processus stochastique $X(t,\zeta)$ est ergodique si toutes les moyennes temporelles existent et ont même valeur pour tout échantillon.

Un processus est dit à moyenne ergodique si :

$$E[X(t,\zeta)] = m_X(t) =< X(t,\zeta) >$$

Un processus est dit à autocorrélation ergodique si :

$$\gamma_X(t,\zeta) =< X(t,\zeta)X(t+\tau,\zeta) >= \overline{\gamma}_X(t,\zeta)$$

6.3 Manipulation

6.3.1 Etude d'un processus aléatoire uniforme et gaussien

Ecrire un programme `MatLab` qui permet de réaliser les opérations suivantes :

1. Générer et représenter graphiquement N points d'un processus aléatoire uniforme de moyenne nulle et de variance σ^2 égale à 1.
2. Générer et représenter graphiquement N points d'un processus aléatoire gaussien de moyenne nulle et de variance σ^2 égale à 1.
3. Tracer et commenter l'histogramme de chaque processus aléatoire.
4. Générer un ensemble de N variables aléatoires $\underline{X}(t)$ gaussiennes indépendantes et identiquement distribuées de variance σ^2 (bruit blanc gaussien).
5. Calculer la matrice de covariance du vecteur $\underline{X}$ correspondant.
6. Estimer la fonction de corrélation de ce processus et comparer à la fonction de corrélation théorique. ? Que concluez-vous ?

6.3.2 Etude des propriétés statistiques d'un signal aléatoire

On considère le signal aléatoire $x(t) = A\cos(\omega_o(t) + \Theta)$, où A et ω_o sont constantes et Θ est une variable aléatoire uniforme sur $[0, \pi]$.

Ecrire un programme `MatLab` qui permet de réaliser les opérations suivantes :

1. Générer et représenter graphiquement N points du signal aléatoire $x(t)$.
2. Calculer la moyenne du signal $E[x(t)]$.

3. Calculer la variance du signal.
4. Calculer la fonction d'autocorrélation.
5. Calculer et représenter graphiquement la densité spectrale de puissance $\Phi_X(f)$.
6. Quelle est la puissance du signal $x(t)$?

Annexe A

Annexe : Les fonctions usuelles de MatLab

A.1 Commandes générales

Lancement et arrêt de MatLab	
matlab.rc	Fichier M de démarrage principal
quit	Permet de quitter Matlab
startup.m	Fichier M de démarrage secondaire

Gestion de l'environnement	
addpath	Ajoute un répertoire à la liste des chemins de recherche de Matlab
doc	Charge la documentions hypertexte
help	Lance l'aide en ligne
lasterr	Fournit le dernier message d'erreur
lookfor	Lance une recherche par mot-clé
path	Affiche la liste des chemins de recherche
profile	Mesure le temps d'exécution d'un fichier M
rmpath	Enlève un répertoire de la liste des chemins de recherche MatLab
type	Liste le contenu d'un fichier
version	Affiche la version de Matlab en cours d'exécution
what	Donne la liste des fichiers M, MEX et MAT présents dans le répertoire courant
whatsnew	Affiche les fichiers readme pour Matlab et ses boîtes à outils
which	Localiser les fichiers et les fonctions

Gestion des variables et de l'espace mémoire	
`clear`	Efface les variables choisies de la mémoire
`disp`	Afficher du texte ou des tableaux
`load`	Charge des variables depuis un fichier
`pack`	Lance la défragmentation de l'espace mémoire de travail
`save`	Sauvegarde des variables dans un fichier
`size`	Fournit les dimensions d'une matrice ou vecteur
`who, whos`	Liste toutes les variables connues de l'espace de travail

Gestion de la fenêtre de commande	
`echo`	Affiche les commandes du fichier M couramment exécuté
`format`	Définit le format d'affichage des nombres
`more`	Active et désactive le mode d'affichage par page

A.2 Opérateurs et caractères spéciaux

Opérateurs et caractères usuels	
+	Addition de réels ou de matrices
−	Soustraction de réels ou de matrices
*	Produit de réels ou de matrices
.*	Produit élément par élément de matrices
\	Division à gauche de réels ou de matrices
/	Division à droite de réels ou de matrices
.\	Division à gauche élément par élément de matrices
./	Division à droite élément par élément de matrices
.	Point décimal
..	Désigne le répertoire parent du répertoire courant
...	Symbole de continuation (pour terminer une instruction ou une expression à la ligne suivante)
%	Commentaire
'	Matrice adjointe (transposée + conjuguée) et délimiteur de chaîne de caractère
.'	Matrice transposée
!	Lance une commande système dont le nom suit, tout en restant sous MatLab
=	Affectation

Constantes et variables spéciales	
`ans`	Dernier résultat calculé
`computer`	Identifie le type d'ordinateur sur lequel est exécuté `Matlab`
`eps`	Précision relative des calculs menés en virgule flottante
`flops`	Compte le nombre d'opérations élémentaires
`i` ou `j`	Racine de -1 (`sqrt(-1)`) nombre imaginaire
`Inf ou inf`	Infini, Par exemple : 10/0
`inputname`	Nom de l'argument en entrée
`NaN ou nan`	Indéterminé, par exemple : 0/0
`nargin`	Nombre d'arguments passés en entrée à une fonction
`nargout`	Nombre d'arguments fournis en sortie par une fonction
`pi`	Désigne la constante trigonométrique $\pi = 3.141592653589793$
`realmax`	Plus grand petit réel positif (double précision)
`realmin`	Plus petit réel positif (double précision)
`varargin`	Liste d'arguments d'entrée contenant les arguments optionnels d'appel d'une fonction
`varargout`	Liste d'arguments de retour contenant les arguments optionnels de retour d'une fonction
`true` ou `1`	Vrai
`false` ou `0`	Faux

Opérateurs relationnels	
`a == b` ou `a eq b`	Test d'égalité qui retourne 1 si a = b et 0 sinon
`isequal(a,b)`	Test d'égalité valable pour des vecteurs et des matrices
`a ~= b` ou `a ne b`	Test de différence qui retourne 0 si a = b et 1 sinon
`a > b` ou `a gt b`	Test de supériorité qui retourne 1 si inégalité vérifiée et 0 sinon
`a < b` ou `a lt b`	Test d'infériorité qui retourne 1 si inégalité vérifiée et 0 sinon
`a >= b`	1 Si inégalité vérifiée et 0 sinon
`a <= b`	1 Si inégalité vérifiée et 0 sinon

Opérateurs logiques	
~`a` ou `not(a)`	Négative logique qui retourne 1 si a = 0 et 0 si a=1
`a & b` ou `and(a,b)`	ET logique (0 ET 0 $\Rightarrow$ 0 ; 0 ET 1 $\Rightarrow$ 0 ; 1 ET 1 $\Rightarrow$1)
`a | b` ou `or(a,b)`	OU logique (0 OU 0 $\Rightarrow$ 0 ; 0 OU 1 $\Rightarrow$ 1 ; 1 OU 1 $\Rightarrow$ 1)
`xor(a,b)`	OU EXCLUSIF (0 OU EXCL 0 $\Rightarrow$ 0 ; 0 OU EXCL 1 $\Rightarrow$ 1 ; 1 OU 1 $\Rightarrow$ 0)

A.3 Matrices et vecteurs remarquables

Matrices et vecteurs remarquables	
`eye`	Matrice identité
`linspace`	Génère un vecteur ligne de valeurs également espacées
`logspace`	Génère un vecteur ligne de valeurs logarithmiquement espacées
`ones`	Matrices composées de 1
`rand`	Matrice (ou nombre) généré aléatoirement selon une distribution uniforme
`randn`	Matrice (ou nombre) généré aléatoirement selon une distribution normale
`zeros`	Matrice nulle

A.4 Opérations sur les vecteurs et les matrices

Opérations sur les vecteurs et les matrices	
`cat`	Permet la concaténation de tableaux
`diag`	Permet la définition de matrices diagonales ou l'extraction de la diagonale d'une matrice
`fliplr`	Permutation circulaire des colonnes
`flipud`	Permutation circulaire des lignes
`repmat`	Permet la réplication d'une valeur dans une matrice
`reshape`	Permet de transformer et redimensionner une matrice
`rot90`	Transformation équivalente à l'application successive de la transposition et de flipud
`tril`	Extrait le triangle inférieur d'une matrice
`triu`	Extrait le triangle supérieur d'une matrice
`numel(M)`	Nombre d'éléments de matrice M
`size(M)`	Taille d'une matrice (nb lignes × nb colonnes)
`rank(M)`	Rang d'une matrice (nb de lignes ou colonnes linéairement indépendants)
`trnspose(M) = M'`	Transposée d'une matrice
`M1+M2, M1-M2`	Addition et soustraction de deux matrices (`!dim(M1) = dim(M2)`)

α*M	Multiplication d'une matrice par scalaire α
M1*M2 ou mtimes	Multiplication de deux matrices (nb colonnes M1 = nb colonne M2)
M1.M2 ou times	Multiplication élément par élément (!dim(M1) = dim(M2))
inv(M) ou M^-1	Inversion d'une matrice (M = matrice carrée)
det(M)	Déterminant d'une matrice
trace(M)	Trace d'une matrice
[P, D]=eig(M)	Valeurs et vecteurs propres d'une matrice
mean(M)	Moyenne d'une matrice
std(M)	Ecart-type d'une matrice
[A,B,C]	Juxtaposition horizontale des matrices A, B et C
[A;B;C]	Juxtaposition verticale des matrices A, B et C
N=0*J	Astuce pour obtenir une matrice N nulle de même taille que J sans connaître la taille de J
sparse	Création d'une matrice «creuse» (cf help sparse)
length(V)	Dimension (ou nombre d'éléments) du vecteur V
min(V)	Minimum d'un vecteur
max(V)	Maximum d'un vecteur
norm(V)	Norme du vecteur V
mean(V)	Moyenne arithmétique
dot(V1, V2)	Produit scalaire
cross(V1, V2)	Produit vectoriel
expm(M)	Exponentielle matricielle
sqrtm(M)	Racine carrée matricielle (cf. help sqrtm)
logm(M)	Logarithme matriciel (cf. help logm)
M*V	Image du vecteur colonne V par la matrice carrée M
cumsum(V)	Sommes cumulatives des entrées du vecteur V
cumsum(M)	Matrice des sommes cumulatives des colonnes de M
cumprod(V)	Produits cumulatifs des entrées du vecteur V
cumprod(M)	Matrice des produits cumulatifs des colonnes de M
diag	création ou extraction de la diagonale d'une matrice

A.5 Fonctions mathématiques usuelles

Fonctions mathématiques usuelles	
`abs`	Valeur absolue d'un réel ou module d'un complexe
`acos, asin, atan, acot`	Fonctions circulaires réciproques
`acosh, asinh, atanh, acoth`	Fonctions hyperboliques réciproques
`acsc, asec`	Fonctions réciproques de la cosécante et de la séquante circulaires
`acsch, asech`	Fonctions réciproques de la cosécante et de la séquante hyperboliques
`ceil`	Plus petit entier supérieur au nombre flottant considéré
`cos, sin, tan, cot`	Fonctions circulaires directes
`cosh, sinh, tanh, coth`	Fonctions hyperboliques directes
`csc, sec`	Fonctions directes de la cosécante et de la séquante circulaires
`csch, sech`	Fonctions directes de la cosécante et de la séquante hyperboliques
`exp`	Fonction exponentielle népérienne
`fix`	Arrondi entier vers 0
`floor`	Plus grand entier inferieur au nombre flottant considéré (partie entière)
`gcd`	Plus grand commun diviseur
`imag`	Partie imaginaire d'un nombre complexe
`log`	Fonction logarithme népérien
`log2`	Fonction logarithme en base 2
`log10`	Fonction logarithme en base 10
`mod`	Reste (signé) d'une division euclidienne (fonction mudulo)
`rem`	Reste d'une division euclidienne
`round`	Arrondi au plus proche entier
`sign`	Signe
`sqrt`	Racine carrée

A.6 Représentations graphiques

Représentation graphique en dimension 2	
`plot`	Graphique en coordonnées linéaires
`stem`	Graphe d'un signal (style *discret*)
`loglog`	Graphique en coordonnées logarithmiques
`polar`	Graphique en coordonnées polaires
`semilogx, semilogy`	Graphique en coordonnées semi logarithmique
`hist`	Histogramme
`contour`	Trace les lignes de niveau

Représentation graphique en dimension 3	
`fill`	Représentation à base de polygones
`mesh`	Représentation d'une nappe par maillage
`plot3`	Représentation de segments et de points en dimension 3
`surf`	Représentation d'une nappe par une surface
`specgram`	Spectrogramme (utile pour le calcul du périodogramme cumulé)
`psd`	Densité spectrale de puissance
`quantiz`	Quantification uniforme du signal

Gestion des axes	
`axes`	Création d'axes en positions arbitraires
`axis`	Contrôle de l'apparence et de l'échelle des axes
`box`	Encadre le graphique avec des axes
`grid`	Crée un quadrillage superposé au graphique
`hold`	Gèle la figure graphique courante
`subplot`	Juxtapose des graphiques dans la même figure
`zoom`	Agrandit ou diminue une figure

Annotation d'une figure	
`gtext`	Permet de placer du texte sur une figure à l'aide de la souris
`legend`	Ajoute une légende à la figure
`text`	Place du texte sur la figure
`title`	Place un titre sur la figure
`xlabel, ylabel, zlabel`	Mettre une étiquette sur l'axe correspondant

A.7 Analyse de données

Fonctions élémentaires	
convhull	Enveloppe convexe
cumprod	Produit cumulé
cumsum	Somme cumulée
dsearch	Recherche du point le plus proche
median	Valeur médiane d'un tableau
perms	Liste toutes les permutations possibles
prodv	Effectue le produit des éléments d'un tableau
sort	Trie les éléments d'un tableau
sortrows	Trie les lignes d'une matrice par ordre croissant
sum	Effectue la somme des éléments d'un tableau
trapz	Intégration numérique selon la méthode des trapèzes
tsearch	Recherche d'un triangle de Delaunay englobant le point considère
lin2mu	Conversion à la loi μ
mu2lin	Conversion inverse (de la loi μ à la loi linéaire)
dtrend	Centrage d'un signal aléatoire

Filtrage, Corrélation et convolution	
corrcoef	Coefficient de corrélation
cov	Matrice de covariance
xcorr	Autocorrélation
conv	Convolution et produit de polynômes
conv2	Convolution bidimensionnelle
deconv	Déconvolution et division de polynôme
filter	Filtrage numérique avec filtre de type RIF (réponse impulsion-nelle finie) et avec filtres de type RII (réponse impulsionnelle infinie)
filter2	Filtrage numérique bidimensionnel
freqz	Réponse en fréquence d'un filtre
fir1	Conception d'un filtre à réponse impulsionnelle finie
buttord, cheb1ord, cheb2ord, ellipord	Synthèse des filtres numériques

Transformation de Fourier	
`real`	Partie réelle d'un nombre complexe
`imag`	Partie imaginaire d'un nombre complexe
`conj`	Conjugué d'un nombre complexe
`abs`	Module d'un nombre complexe
`angle`	Argument d'un nombre complexe
`fft`	Transformée de Fourier rapide monodimensionnelle
`fft2`	Transformée de Fourier rapide bidimensionnelle
`fftshift`	Permet le centrage du spectre d'une transformée de Fourier rapide
`ifft`	Ttransformée de Fourier inverse monodimensionnelle
`ifft2`	Transformée de Fourier inverse bidimensionnelle
`nextpow2`	Détermine la puissance de 2 immédiatement supérieure au nombre passé en argument
`unwrap`	Corrige les angles des phases